

St. Wilfred's College of Computer Sciences

Affiliated to University of Mumbai, DTE Code: 3539 near MBMC
Garden, Sanghvi Nagar, Mira Road (East), Thane-401107,
Maharashtra



Name: Shah Sahil Gulab

Class: FYMCA

Subject: Advanced Database Management System Lab

Roll No: MCA2025054

Year: 2025-2026



St. WILFRED'S COLLEGE OF COMPUTER SCIENCE

Affiliated to Mumbai University, Approved by AICTE- New Delhi,

Near MBMC garden, Sanghvi Nagar, Mira Bhayandar Road, Mira Road, Thane - 401107

Date:

CERTIFICATE

This is to certify that Miss. Shah Sahil Gulab Roll No. MCA2025054 is a student of FYMCA Semester-I has completed successfully full-semester practical/assignments of subject Advanced Database Management S for the academic year 2025 – 26.

CO	R1 (JOURNAL)	R2 (PERFORMANCE DURING LAB SESSION)	R3 (IMPLEMENTATION USING DIFFERENT PROBLEM SOLVING TECHNIQUES)	R4 (MOCK VIVA)	ATTENDANCE
C01					
C02					
C03					
C04					

Subject In-Charge

Dr. Shubhi Lall Agarwal
Director

External Examiner

Sr.No	PROBLEM STATEMENT	DATE	CO	SIGNATURE
1.	Implementation of Data partitioning through Hash, Range and List partitioning		CO1	
2.	Implementation of Analytical queries like Roll_UP, CUBE, First, Last, Lead, Lag, Rank AND Dense Rank and Windowing functions- preceding rows, following and rows between		CO3	
3.	Implementation of ORDBMS concepts like ADT (Abstract Data Types), Object table and Inheritance		CO1	
<u>4.</u>	ETL Transformation with Pentaho 1. Copy data from Source & store to target 2. Adding Sequence 3. Adding Calculator 4. Concatenation of Two Fields 5. Splitting of Two Fields 6. Number Range 7. String Operations 8. Sorting Data 9. Implement the Merge Join 10. Implement data validations on table data 11. Replace Strings 12. Splitting Fields to Rows		CO2	
<u>5.</u>	Introduction to R , Install packages, Loading packages Data types, checking variable type, printing variable and objects (Vector, Matrix, List, Factor, Data frame, Table) c-binding and rbinding Reading and Writing data Setwd(), getwd(), data(), rm() Attaching and Detaching data Reading data from the console Loading data from different data sources (CSV, Excel)		CO4	
<u>6.</u>	Data preprocessing techniques in R Naming and Renaming variables Adding a new variables Dealing with missing value Dealing with categorical data Data reduction using subsetting		CO4	
<u>7.</u>	Implementation and analysis of Linear regression.		CO4	
<u>8.</u>	Implementation and Analysis Classification algorithms -Naïve Bayesian		CO5	

<u>9.</u>	Implementation and Analysis Classification algorithms - K-Nearest Neighbour		CO5	
<u>10.</u>	Implementation and Analysis Classification algorithms - ID3, C4.5		CO5	
<u>11.</u>	Implementation and analysis of Apriori Algorithm using Market Basket Analysis		CO5	
<u>12.</u>	Implementation and analysis of clustering algorithms - K-Means		CO5	
<u>13.</u>	Implementation and analysis of clustering algorithms - Agglomerative		CO5	

PRACTICAL NO 1

Aim: Implementation of Data partitioning through Range , List And Hash partitioning a) Range partitioning

```
create table Accounts
(
acctno          number,
custname varchar(23), branch
               varchar(23), acctbal number
)partition by range (acctbal)
(
partition p1 values less than (2000), partition
p2 values less than (5000),
partition p3 values less than (9999)
);
select * from Accounts; Output:
```

```
SQL> describe Accounts
```

Name	Null?	Type
ACCOUNTNO		NUMBER
CUSTOMERNAME		VARCHAR2(23)
BRANCH		VARCHAR2(23)
ACCOUNTBAL		NUMBER

```
SQL> insert into Accounts values(4, 'Aarti','Airport',5000);
```

```
1 row created.
```

```
SQL> insert into Accounts values(1, 'Rohan', 'Thane', 4000);
```

```
1 row created.
```

```
SQL> insert into Accounts values(5, 'Aayush', 'Kalyan', 1000);
```

```
1 row created.
```

```
SQL> insert into Accounts values(7, 'Pratik', 'Mumbai', 7000);
```

```
1 row created.
```

```
SQL> insert into Accounts values(3, 'Sagar', 'Fun Fiesta', 6000);
```

```
1 row created.
```

```
SQL>
```

```
SQL> select *from Accounts;
```

ACCOUNTNO	CUSTOMERNAME	BRANCH	ACCOUNTBAL
5	Aayush	Kalyan	1000
1	Rohan	Thane	4000
4	Aarti	Airport	5000
7	Pratik	Mumbai	7000
3	Sagar	Fun Fiesta	6000

```
SQL> |
```

```
select * from Accounts partition (p2),
```

```
select * from Accounts partition (p3); Output:
```

```
SQL> select *from Accounts;
```

ACCOUNTNO	CUSTOMERNAME	BRANCH	ACCOUNTBAL
5	Aayush	Kalyan	1000
1	Rohan	Thane	4000
4	Aarti	Airport	5000
7	Pratik	Mumbai	7000
3	Sagar	Fun Fiesta	6000

```
SQL> select *from Accounts partition(p1);
```

ACCOUNTNO	CUSTOMERNAME	BRANCH	ACCOUNTBAL
5	Aayush	Kalyan	1000

```
SQL> select *from Accounts partition(p2);
```

ACCOUNTNO	CUSTOMERNAME	BRANCH	ACCOUNTBAL
1	Rohan	Thane	4000

```
SQL> select *from Accounts partition(p3);
```

ACCOUNTNO	CUSTOMERNAME	BRANCH	ACCOUNTBAL
4	Aarti	Airport	5000
7	Pratik	Mumbai	7000
3	Sagar	Fun Fiesta	6000

```
SQL> |
```

alter table Accounts merge Partitions p1,p2 into partition p6; select *
from Accounts partition(p6); **Output:**

```
SQL> alter table Accounts merge Partitions p1, p2 into partition p6;
```

Table altered.

```
SQL> select *from Accounts partition(p6);
```

ACCOUNTNO	CUSTOMERNAME	BRANCH	ACCOUNTBAL
5	Aayush	Kalyan	1000
1	Rohan	Thane	4000

```
SQL>
```

```
SQL> alter table Accounts add partition p4 values less than(10000);
```

Table altered.

```
SQL> alter table Accounts add partition p5 vauess less than(120000);  
alter table Accounts add partition p5 vauess less than(120000)  
*
```

ERROR at line 1:
ORA-14020: this physical attribute may not be specified for a table partition

```
SQL> alter table Accounts add partition p5 values less than(12000);
```

Table altered.

```
SQL> insert into Accounts values(2, 'Ziya', 'Mahim', 11000);
```

1 row created.

```
SQL> insert into Accounts values(9, 'Sairaj', 'Airpot', 105000);  
insert into Accounts values(9, 'Sairaj', 'Airpot', 105000)  
*
```

ERROR at line 1:
ORA-14400: inserted partition key does not map to any partition

```
SQL> insert into Accounts values(9, 'Sairaj', 'Airport', 10500);
```

1 row created.

```
SQL> select *from Accounts partition(p5);
```

ACCOUNTNO	CUSTOMERNAME	BRANCH	ACCOUNTBAL
2	Ziya	Mahim	11000
9	Sairaj	Airport	10500

```
SQL> |
```

```
SQL> SELECT * FROM USER_TAB_PARTITIONS WHERE TABLE_NAME = 'ACCOUNTS';
```

TABLE_NAME	COM	PARTITION_NAME
ACCOUNTS	NO	P6
SUBPARTITION_COUNT		
HIGH_VALUE		
HIGH_VALUE_LENGTH	PARTITION_POSITION	TABLESPACE_NAME
PCT_USED	INI_TRANS	MAX_TRANS
INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENT
MAX_EXTENT	PCT_INCREASE	FREELISTS
FREELIST_GROUPS	LOGGING	COMPRESS
NUM_ROWS	BLOCKS	EMPTY_BLOCKS
AUG_SPACE	CHAIN_CNT	AUG_ROW_LEN
SAMPLE_SIZE	LAST_ANAL	
BUFFER_	GLO	USE
ACCOUNTS	NO	P6
TABLE_NAME	COM	PARTITION_NAME
SUBPARTITION_COUNT		
HIGH_VALUE		
HIGH_VALUE_LENGTH	PARTITION_POSITION	TABLESPACE_NAME
PCT_USED	INI_TRANS	MAX_TRANS
INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENT
MAX_EXTENT	PCT_INCREASE	FREELISTS
FREELIST_GROUPS	LOGGING	COMPRESS
NUM_ROWS	BLOCKS	EMPTY_BLOCKS
AUG_SPACE	CHAIN_CNT	AUG_ROW_LEN
SAMPLE_SIZE	LAST_ANAL	
BUFFER_	GLO	USE
		0
TABLE_NAME	COM	PARTITION_NAME
SUBPARTITION_COUNT		

b) List Partitioning

```
SQL> create table Product(
2  prdno varchar(10),
3  prodname varchar(23),
4  rate number,
5  qty_available number
6  )
7  partition by list(prdno)
8  (
9  partition pd1 values('p001','p002','p003','p004'),
10 partition pd2 values('b001','b002','b003','b004'),
11 partition pd3 values('c001','c002','c003','c004'),
12 partition pd4 values('d001','d002','d003','d004')
13 );
```

Table created.

```
SQL> desc Product;
```

Name	Null?	Type
PRDNO		VARCHAR2(10)
PRODNAME		VARCHAR2(23)
RATE		NUMBER
QTY_AVAILABLE		NUMBER

```

SQL>
SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('p001', 'Desktop Monitor 27"', 250.00, 150);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('p002', 'Wireless Keyboard', 75.50, 420);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('p003', 'Gaming Mouse', 45.99, 310);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('p004', 'Webcam Pro HD', 89.95, 255);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('b001', 'External SSD 1TB', 120.00, 95);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('b002', 'USB-C Hub 7-in-1', 59.90, 180);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('b003', 'Laptop Cooling Pad', 35.00, 115);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('b004', 'Noise Cancelling Headph', 199.99, 130);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('c001', 'Smart Watch Series 9', 450.00, 50);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('c002', 'Fitness Tracker Lite', 79.00, 200);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('c003', 'Portable Bluetooth Spkr', 65.40, 300);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('c004', 'VR Headset Entry', 299.00, 75);

1 row created.

```

```

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('c004', 'VR Headset Entry', 299.00, 75);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('d001', '4K HDMI Cable', 15.00, 500);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('d002', 'Magnetic Phone Mount', 22.50, 650);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('d003', 'Travel Router', 55.00, 190);

1 row created.

SQL> INSERT INTO Product (prdno, prodname, rate, qty_available)
  2  VALUES ('d004', 'Power Bank 20000mAh', 49.99, 210);

SQL> |

```

PRDNO	PRODNAME	RATE	QTY_AVAILABLE
p001	Desktop Monitor 27"	250	150
p002	Wireless Keyboard	75.5	420
p003	Gaming Mouse	45.99	310
p004	Webcam Pro HD	89.95	255
b001	External SSD 1TB	120	95
b002	USB-C Hub 7-in-1	59.9	180
b003	Laptop Cooling Pad	35	115
b004	Noise Cancelling Headph	199.99	130
c001	Smart Watch Series 9	450	50
c002	Fitness Tracker Lite	79	200
c003	Portable Bluetooth Spkr	65.4	300

PRDNO	PRODNAME	RATE	QTY_AVAILABLE
c004	VR Headset Entry	299	75
d001	4K HDMI Cable	15	500
d002	Magnetic Phone Mount	22.5	650
d003	Travel Router	55	190

15 rows selected.

SQL>

```
SQL> SELECT *
      2 FROM user_tab_partitions
      3 WHERE table_name = 'PRODUCT';
```

TABLE_NAME	COM PARTITION_NAME
SUBPARTITION_COUNT	
HIGH_VALUE	
HIGH_VALUE_LENGTH PARTITION_POSITION TABLESPACE_NAME	PCT_FREE
PCT_USED INI_TRANS MAX_TRANS INITIAL_EXTENT NEXT_EXTENT MIN_EXTENT	
MAX_EXTENT PCT_INCREASE FREELISTS FREELIST_GROUPS LOGGING COMPRESS NUM_ROWS	
BLOCKS EMPTY_BLOCKS AVG_SPACE CHAIN_CNT AVG_ROW_LEN SAMPLE_SIZE LAST_ANAL	
BUFFER_GLO USE	
PRODUCT	NO PD1

TABLE_NAME	COM PARTITION_NAME
SUBPARTITION_COUNT	
HIGH_VALUE	
HIGH_VALUE_LENGTH PARTITION_POSITION TABLESPACE_NAME	PCT_FREE
PCT_USED INI_TRANS MAX_TRANS INITIAL_EXTENT NEXT_EXTENT MIN_EXTENT	
MAX_EXTENT PCT_INCREASE FREELISTS FREELIST_GROUPS LOGGING COMPRESS NUM_ROWS	
BLOCKS EMPTY_BLOCKS AVG_SPACE CHAIN_CNT AVG_ROW_LEN SAMPLE_SIZE LAST_ANAL	
BUFFER_GLO USE	
	0

TABLE_NAME	COM PARTITION_NAME
SUBPARTITION_COUNT	
HIGH_VALUE	
HIGH_VALUE_LENGTH PARTITION_POSITION TABLESPACE_NAME	PCT_FREE
PCT_USED INI_TRANS MAX_TRANS INITIAL_EXTENT NEXT_EXTENT MIN_EXTENT	
MAX_EXTENT PCT_INCREASE FREELISTS FREELIST_GROUPS LOGGING COMPRESS NUM_ROWS	
BLOCKS EMPTY_BLOCKS AVG_SPACE CHAIN_CNT AVG_ROW_LEN SAMPLE_SIZE LAST_ANAL	
BUFFER_GLO USE	

```

SQL> insert into Product(prdno, prodname, rate, qty_available) values( 'f001', 'Pouch', 150.00, 5);
insert into Product(prdno, prodname, rate, qty_available) values( 'f001', 'Pouch', 150.00, 5)
*
ERROR at line 1:
ORA-00904: "QTY_AVAILABLE": invalid identifier

SQL> insert into Product(prdno, prodname, rate, qty_available) values( 'f001', 'Pouch', 150.00, 5)
;
1 row created.

SQL> insert into Product(prdno, prodname, rate, qty_available) values( 'f002', 'Bottle', 200.00, 10)
);
1 row created.

SQL> insert into Product(prdno, prodname, rate, qty_available) values( 'f003', 'Moniter', 7000.00,
15);
1 row created.

SQL> insert into Product(prdno, prodname, rate, qty_available) values( 'f004', 'Power Bank', 1000.0
0, 20);
1 row created.

SQL> select *from Product Partition(pd5);

```

PRDNO	PRODNAME	RATE	QTY_AVAILABLE
f001	Pouch	150	5
f002	Bottle	200	10
f003	Moniter	7000	15
f004	Power Bank	1000	20

```

con \ 1

```

```
SQL> alter table Product rename partition pd5 to pd6;
```

Table altered.

```
SQL> select *from Product Partition(pd6);
```

PRDNO	PRODNAME	RATE	QTY_AVAILABLE
f001	Pouch	150	5
f002	Bottle	200	10
f003	Moniter	7000	15
f004	Power Bank	1000	20

```
SQL>
```

```
SQL> select *from Product Partition(pd6);
```

PRDNO	PRODNAME	RATE	QTY_AVAILABLE
f001	Pouch	150	5
f002	Bottle	200	10
f003	Moniter	7000	15
f004	Power Bank	1000	20

```
SQL> alter table PRODUCT Modify partition PD6 add values('f005');
```

Table altered.

```
SQL> select *from Product Partition(pd6);
```

PRDNO	PRODNAME	RATE	QTY_AVAILABLE
f001	Pouch	150	5
f002	Bottle	200	10
f003	Moniter	7000	15
f004	Power Bank	1000	20

```
SQL> insert into Product(prdno, prodname, rate, qty_available) values('F005', 'Phone', 20000.00, 25)
;
```

1 row created.

```
SQL> select *from Product Partition(pd6);
```

PRDNO	PRODNAME	RATE	QTY_AVAILABLE
F001	Pouch	150	5
F002	Bottle	200	10
F003	Moniter	7000	15
F004	Power Bank	1000	20
F005	Phone	20000	25

```
SQL>
```

```
SQL> delete from Product where prdno ='F005';
```

1 row deleted.

```
SQL> select *from Product Partition(pd6);
```

PRDNO	PRODNAME	RATE	QTY_AVAILABLE
F001	Pouch	150	5
F002	Bottle	200	10
F003	Moniter	7000	15
F004	Power Bank	1000	20

```
SQL> |
```

c) Hash Partitioning

```
SQL> create table employees
2 (empno number,
3 empname varchar(30),
4 department varchar(30),
5 salary number)
6 partition by hash(empno)
7 (partition p1,
8 partition p2,
9 partition p3,
10 partition p4);
```

Table created.

```
SQL> desc employees;
```

Name	Null?	Type
EMPNO		NUMBER
EMPNAME		VARCHAR2(30)
DEPARTMENT		VARCHAR2(30)
SALARY		NUMBER

```
SQL>
```

```

SQL> insert into employees values(1, 'Alisha', 'hr', 4000);
1 row created.
SQL> insert into employees values(2, 'John', 'engineering', 6000);
1 row created.
SQL>
SQL> insert into employees values(3, 'Sonali', 'marketing', 3000);
1 row created.
SQL> insert into employees values((4, 'Sam', 'finance', 7000);
insert into employees values((4, 'Sam', 'finance', 7000)
*
ERROR at line 1:
ORA-00907: missing right parenthesis

SQL> insert into employees values(4, 'Sam', 'finance', 7000);
1 row created.
SQL> insert into employees values(5, 'Vicky', 'hr', 2000);
1 row created.
SQL> insert into employees values(6, 'Feriha', 'engineering', 5000);
1 row created.
SQL> |

```

```
SQL> select *from employees;
```

EMPNO	EMPNAME	DEPARTMENT
6	Feriha	engineering
5000		
2	John	engineering
6000		
5	Vicky	hr
2000		
1	Alisha	hr
4000		
3	Sonali	marketing
3000		
4	Sam	finance
7000		

```
6 rows selected.
```

```
SQL> select *from employees partition(p1);
```

EMPNO	EMPNAME	DEPARTMENT
6	Feriha	engineering
5000		

```
SQL> select *from employees partition(p2);
```

```
no rows selected
```



```
SQL> select *from employees partition(p3);
```

EMPNO	EMPNAME	DEPARTMENT

SALARY		

2	John	engineering
6000		
5	Vicky	hr
2000		

```
SQL> select *from employees partition(p4);
```

EMPNO	EMPNAME	DEPARTMENT

SALARY		

1	Alisha	hr
4000		
3	Sonali	marketing
3000		
4	Sam	finance
7000		

```
SQL> |
```

```
SQL> update employees set salary = salary*(salary*0.05) where salary<5000;
```

```
3 rows updated.
```

```
SQL> select *from employees;
```

EMPNO	EMPNAME	DEPARTMENT

SALARY		

6	Feriha	engineering
5000		
2	John	engineering
6000		
5	Vicky	hr
2100		

EMPNO	EMPNAME	DEPARTMENT

SALARY		

1	Alisha	hr
4200		
3	Sonali	marketing
3150		
4	Sam	finance
7000		

```
6 rows selected.
```

```
SQL>
```

PRACTICAL NO 2

AIM : Implementation of Analytical queries like Roll_UP, CUBE, First, Last, Lead, Lag, Rank AND Dense Rank and Windowing functions- preceding rows, following and rows between

```
create table employee1
(
  empno      number,
  empname    varchar(30),
  department varchar(30), salary
  number
);
```

```
SQL> create table employee1
2  (
3  empno number,
4  empname varchar(30),
5  department varchar(30),
6  salary number
7  );
```

Table created.

```
SQL> select * from employee1;
```

```
SQL> select * from employee1;
```

no rows selected

```
SQL> desc employee1;
```

Name	Null?	Type
EMPNO		NUMBER
EMPNAME		VARCHAR2(30)
DEPARTMENT		VARCHAR2(30)
SALARY		NUMBER

```
SQL> insert into employee1 values (1, 'Siddharth', 'hr', 4000);
```

```
SQL> insert into employee1 values (2, 'Sarvesh', 'engineering', 6000);
```

```
SQL> insert into employee1 values (3, 'Ziya', 'marketing', 3000);
```

```
SQL> insert into employee1 values (4, 'Vanshika', 'finance', 7000);
```

```
SQL> insert into employee1 values (5, 'Akash', 'hr', 2000);
```

```
SQL> insert into employee1 values (6, 'Pratik', 'engineering', 5000);
```

```
SQL> insert into employee1 values (7, 'Tanveer', 'engineering', 6000);
```

```
SQL> insert into employee1 values (8, 'Amey', 'marketing', 3000);
```

```
SQL> insert into employee1 values (9, 'Sarah', 'finance', 7000);
```

```
SQL> insert into employee1 values (10, 'Virendra', 'finance', 2000);
```

```

SQL> insert into employee1 values (1, 'Siddharth', 'hr', 4000);
1 row created.
SQL> insert into employee1 values (2, 'Sarvesh', 'engineering', 6000);
1 row created.
SQL> insert into employee1 values (3, 'Ziya', 'marketing', 3000);
1 row created.
SQL> insert into employee1 values (4, 'Vanshika', 'finance', 7000);
1 row created.
SQL> insert into employee1 values (5, 'Akash', 'hr', 2000);
1 row created.
SQL> insert into employee1 values (6, 'Pratik', 'engineering', 5000);
1 row created.
SQL> insert into employee1 values (7, 'Tanveer', 'engineering', 6000);
1 row created.
SQL> insert into employee1 values (8, 'Amey', 'marketing', 3000);
1 row created.
SQL> insert into employee1 values (9, 'Sarah', 'finance', 7000);
1 row created.
SQL> insert into employee1 values (10, 'Virendra', 'finance', 2000);
1 row created.

```

```

SQL> set pagesize 2000;
SQL> set linesize 1000;

```

```

SQL> select * from employee1;

```

EMPNO	EMPNAME	DEPARTMENT	SALARY
1	Siddharth	hr	4000
2	Sarvesh	engineering	6000
3	Ziya	marketing	3000
4	Vanshika	finance	7000
5	Akash	hr	2000
6	Pratik	engineering	5000
7	Tanveer	engineering	6000
8	Amey	marketing	3000
9	Sarah	finance	7000
10	Virendra	finance	2000

10 rows selected.

Rank select Empno, empname, department ,salary, rank() over (order by salary) "Rank" from employee1;

```

SQL> select Empno, empname, department ,salary, rank() over (order by salary) "Rank" from employee1;

```

EMPNO	EMPNAME	DEPARTMENT	SALARY	Rank
5	Akash	hr	2000	1
10	Virendra	finance	2000	1
3	Ziya	marketing	3000	3
8	Amey	marketing	3000	3
1	Siddharth	hr	4000	5
6	Pratik	engineering	5000	6
7	Tanveer	engineering	6000	7
2	Sarvesh	engineering	6000	7
4	Vanshika	finance	7000	9
9	Sarah	finance	7000	9

10 rows selected.

Dense Rank select empno, empname, department, salary, dense_rank() over(order by salary)"den_rank"
from employee1;

SQL> select empno, empname, department, salary, dense_rank() over(order by salary)"den_rank" from employee1;

EMPNO	EMPNAME	DEPARTMENT	SALARY	den_rank
5	Akash	hr	2000	1
10	Virendra	finance	2000	1
3	Ziya	marketing	3000	2
8	Amev	marketing	3000	2
1	Siddharth	hr	4000	3
6	Pratik	engineering	5000	4
7	Tanveer	engineering	6000	5
2	Sarvesh	engineering	6000	5
4	Vanshika	finance	7000	6
9	Sarah	finance	7000	6

10 rows selected.

cube

select empno, department, sum(salary) from employee1 Group by cube (empno, department) order by
department , empno;

SQL> select empno, department, sum(salary) from employee1 Group by cube (empno, department) order by
department , empno;

EMPNO	DEPARTMENT	SUM(SALARY)
2	engineering	6000
6	engineering	5000
7	engineering	6000
	engineering	17000
4	finance	7000
9	finance	7000
10	finance	2000
	finance	16000
1	hr	4000
5	hr	2000
	hr	6000
3	marketing	3000
8	marketing	3000
	marketing	6000
1		4000
2		6000
3		3000
4		7000
5		2000
6		5000
7		6000
8		3000
9		7000
10		2000
		45000

25 rows selected.

rollup

select empno, department, sum(salary) from employee1 Group by rollup(empno, department) order by
empno, department;

SQL> select empno, department, sum(salary) from employee1 Group by rollup(empno, department) order b
y empno, department;

EMPNO	DEPARTMENT	SUM(SALARY)
1	hr	4000
1		4000
2	engineering	6000
2		6000
3	marketing	3000
3		3000
4	finance	7000
4		7000
5	hr	2000
5		2000
6	engineering	5000
6		5000
7	engineering	6000
7		6000
8	marketing	3000
8		3000
9	finance	7000
9		7000
10	finance	2000
10		2000
		45000

21 rows selected.

```
SQL> select department, last_value(salary) over (partition by department order by salary) as last_salary from employee1;
```

DEPARTMENT	LAST_SALARY
engineering	5000
engineering	6000
engineering	6000
finance	2000
finance	7000
finance	7000
hr	2000
hr	4000
marketing	3000
marketing	3000

last 10 rows selected.

first

```
select department, last_value(salary) over (partition by department order by salary) as last_salary from employee1;
```

```
select department, first_value(salary) over (partition by department order by salary) as first_salary from employee1;
```

```
SQL> select department, first_value(salary) over (partition by department order by salary) as first_salary from employee1;
```

DEPARTMENT	FIRST_SALARY
engineering	5000
engineering	5000
engineering	5000
finance	2000
finance	2000
finance	2000
hr	2000
hr	2000
marketing	3000
marketing	3000

10 rows selected.

lead

```
select empname, salary, lead(salary, 1) over (order by salary asc) as next_salary from employee1;
```

```
SQL> select empname, salary, lead(salary, 1) over (order by salary asc) as next_salary from employee1;
```

EMPNAME	SALARY	NEXT_SALARY
Akash	2000	2000
Virendra	2000	3000
Ziya	3000	3000
Ameey	3000	4000
Siddharth	4000	5000
Pratik	5000	6000
Tanveer	6000	6000
Sarvesh	6000	7000
Vanshika	7000	7000
Sarah	7000	

10 rows selected.

lag

```
select empname, salary, lag(salary, 1) over (order by salary asc) as previous_salary from employee1;
```

```
SQL> select empname, salary, lag(salary, 1) over (order by salary asc) as previous_salary from employee1;
```

EMPNAME	SALARY	PREVIOUS_SALARY
Akash	2000	
Virendra	2000	2000
Ziya	3000	2000
Ameey	3000	3000
Siddharth	4000	3000
Pratik	5000	4000
Tanveer	6000	5000
Sarvesh	6000	6000
Vanshika	7000	6000
Sarah	7000	7000

10 rows selected.

Window

```
CREATE TABLE Employee2(  
    e_id number(4),  
    first_name varchar2(15),  
    last_name varchar2(15),  
    dept_id number(3),  
    hire_date date,  
    salary number(10,2)  
);
```

```
SQL> CREATE TABLE Employee2(  
2     e_id number(4),  
3     first_name varchar2(15),  
4     last_name varchar2(15),  
5     dept_id number(3),  
6     hire_date date,  
7     salary number(10,2)  
8     );
```

Table created.

```
INSERT INTO Employee2 values(101, 'Susmita', 'Raul', 10, '19-Oct-2004', 20000);  
INSERT INTO Employee2 values(107, 'Aarti', 'Chauhan', 10, '04-May-2004', 30000);  
INSERT INTO Employee2 values(106, 'Vedshree', 'Jadhav', 10, '17-March-2004', 50000);  
INSERT INTO Employee2 values(102, 'Stuti', 'Upadhaya', 10, '26-July-2004', 200000);  
INSERT INTO Employee2 values(103, 'Sahil', 'Shah', 10, '19-June-2005', 70000);  
INSERT INTO Employee2 values(104, 'Harshada', 'Deshmukh', 10, '30-November-2004', 40000);  
INSERT INTO Employee2 values(115, 'Prasanth', 'Patil', 20, '16-Jul-2001', 6000);  
INSERT INTO Employee2 values(126, 'Nataraj', 'Patil', 40, '14-Jun-2001', 16000);  
INSERT INTO Employee2 values(116, 'Naresh', 'Patil', 20, '29-Jan-2001', 25000);  
INSERT INTO Employee2 values(118, 'Sreenivas', 'Patil', 29, '29-Jan-2001', 20000);  
INSERT INTO Employee2 values(108, 'Mohammed', 'Singh', 50, '19-Jan-2001', 30000);  
INSERT INTO Employee2 values(110, 'Regina', 'Patil', 30, '16-Feb-2001', 25000);  
INSERT INTO Employee2 values(113, 'Chandan', 'Patil', 30, '10-Mar-2001', 29000);  
Commit;
```

```
SQL> INSERT INTO Employee2 values(101, 'Susmita', 'Raul', 10, '19-Oct-2004', 20000);
```

1 row created.

```
SQL>
```

```
SQL> INSERT INTO Employee2 values(107, 'Aarti', 'Chauhan', 10, '04-May-2004', 30000);
```

1 row created.

```
SQL>
```

```
SQL> INSERT INTO Employee2 values(106, 'Vedshree', 'Jadhav', 10, '17-March-2004', 50000);
```

1 row created.

```
SQL>
```

```
SQL>
```

```
SQL> INSERT INTO Employee2 values(102, 'Stuti', 'Upadhaya', 10, '26-July-2004', 200000);
```

1 row created.

```
SQL>
```

```
SQL> INSERT INTO Employee2 values(103, 'Sahil', 'Shah', 10, '19-June-2005', 70000);
```

1 row created.

```
SQL>
```

```
SQL> INSERT INTO Employee2 values(104, 'Harshada', 'Deshmukh', 10, '30-November-2004', 40000);
```

1 row created.

```

SQL> INSERT INTO Employee2 values(115, 'Prasanth', 'Patil', 20, '16-Jul-2001', 6000);
1 row created.

SQL>
SQL>
SQL> INSERT INTO Employee2 values(126, 'Nataraj', 'Patil', 40, '14-Jun-2001', 16000);
1 row created.

SQL> INSERT INTO Employee2 values(116, 'Naresh', 'Patil', 20, '29-Jan-2001', 25000);
1 row created.

SQL> INSERT INTO Employee2 values(118, 'Sreenivas', 'Patil', 29, '29-Jan-2001', 20000);
1 row created.

SQL> INSERT INTO Employee2 values(108, 'Mohammed', 'Singh', 50, '19-Jan-2001', 30000);
1 row created.

SQL> INSERT INTO Employee2 values(110, 'Regina', 'Patil', 30, '16-Feb-2001', 25000);
1 row created.

SQL> INSERT INTO Employee2 values(113, 'Chandan', 'Patil', 30, '10-Mar-2001', 29000);
1 row created.

SQL> Commit;
Commit complete.

```

```

SELECT e_id,first_name,dept_id,salary,SUM(salary)
OVER (order by e_id)"Cumulative_Sal"
FROM Employee2;

```

```

SQL> set linesize 1000;
SQL> set pagesize 1000;
SQL> SELECT e_id,first_name,dept_id,salary,SUM(salary)
2  OVER (order by e_id)"Cumulative_Sal"
3  FROM Employee2;

```

E_ID	FIRST_NAME	DEPT_ID	SALARY	Cumulative_Sal
101	Susmita	10	20000	20000
102	Stuti	10	200000	220000
103	Sahil	10	70000	290000
104	Harshada	10	40000	330000
106	Vedshree	10	50000	380000
107	Aarti	10	30000	410000
108	Mohammed	50	30000	440000
110	Regina	30	25000	465000
113	Chandan	30	29000	494000
115	Prasanth	20	6000	500000
116	Naresh	20	25000	525000
118	Sreenivas	29	20000	545000
126	Nataraj	40	16000	561000

13 rows selected.

```

SELECT e_id, first_name, dept_id, salary, SUM(salary)
OVER (partition by dept_id order by salary Rows unbounded preceding) "Cumulative" FROM Employee2;

```

```
SQL> SELECT e_id, first_name, dept_id, salary, SUM(salary)
2 OVER (partition by dept_id order by salary Rows unbounded preceding) "Cumulative" FROM Employee
e2;
```

E_ID	FIRST_NAME	DEPT_ID	SALARY	Cumulative
101	Susmita	10	20000	20000
107	Aarti	10	30000	50000
104	Harshada	10	40000	90000
106	Vedshree	10	50000	140000
103	Sahil	10	70000	210000
102	Stuti	10	200000	410000
115	Prasanth	20	6000	6000
116	Naresh	20	25000	31000
118	Sreenivas	29	20000	20000
110	Regina	30	25000	25000
113	Chandan	30	29000	54000
126	Nataraj	40	16000	16000
108	Mohammed	50	30000	30000

13 rows selected.

SELECT e_id, first_name, dept_id, salary,

SUM(salary) OVER (ORDER BY e_id ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
"Cumulative_Sal"

FROM Employee2;

```
SQL> SELECT e_id, first_name, dept_id, salary,
2 SUM(salary) OVER (ORDER BY e_id ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) "Cumulative_Sal"
3 FROM Employee2;
```

E_ID	FIRST_NAME	DEPT_ID	SALARY	Cumulative_Sal
101	Susmita	10	20000	20000
102	Stuti	10	200000	220000
103	Sahil	10	70000	290000
104	Harshada	10	40000	310000
106	Vedshree	10	50000	160000
107	Aarti	10	30000	120000
108	Mohammed	50	30000	110000
110	Regina	30	25000	85000
113	Chandan	30	29000	84000
115	Prasanth	20	6000	60000
116	Naresh	20	25000	60000
118	Sreenivas	29	20000	51000
126	Nataraj	40	16000	61000

13 rows selected.

SELECT e_id, first_name, dept_id, salary,

SUM(salary) OVER (ORDER BY dept_id ROWS BETWEEN 3 PRECEDING and 1 following)
"Cumulative_Sal"

FROM Employee2;

```
SQL> SELECT e_id, first_name, dept_id, salary,
2 SUM(salary) OVER (ORDER BY dept_id ROWS BETWEEN 3 PRECEDING and 1 following) "Cumulative_Sal"
3 FROM Employee2;
```

E_ID	FIRST_NAME	DEPT_ID	SALARY	Cumulative_Sal
101	Susmita	10	20000	50000
107	Aarti	10	30000	100000
106	Vedshree	10	50000	300000
102	Stuti	10	200000	370000
103	Sahil	10	70000	390000
104	Harshada	10	40000	366000
115	Prasanth	20	6000	341000
116	Naresh	20	25000	161000
118	Sreenivas	29	20000	116000
110	Regina	30	25000	105000
113	Chandan	30	29000	115000
126	Nataraj	40	16000	120000
108	Mohammed	50	30000	100000

13 rows selected.

SELECT e_id,first_name,dept_id,salary, SUM(salary)

OVER (order by e_id Rows between 2 preceding and 2 following)"Cumulative_Sal" FROM Employee2;

```
SQL> SELECT e_id,first_name,dept_id,salary, SUM(salary)
2 OVER (order by e_id Rows between 2 preceding and 2 following)"Cumulative_Sal" FROM Employee2;
```

E_ID	FIRST_NAME	DEPT_ID	SALARY	Cumulative_Sal
101	Susmita	10	20000	290000
102	Stuti	10	200000	330000
103	Sahil	10	70000	380000
104	Harshada	10	40000	390000
106	Uedshree	10	50000	220000
107	Aarti	10	30000	175000
108	Mohammed	50	30000	164000
110	Regina	30	25000	120000
113	Chandan	30	29000	115000
115	Prasanth	20	6000	105000
116	Naresh	20	25000	96000
118	Sreenivas	29	20000	67000
126	Nataraj	40	16000	61000

13 rows selected.

SELECT e_id,first_name,hire_date,count(e_id)

OVER(order by hire_date range interval '15' day preceding)"Days" FROM Employee2;

```
SQL> SELECT e_id,first_name,hire_date,count(e_id)
2 OVER(order by hire_date range interval '15' day preceding)"Days"
3 FROM Employee2;
```

E_ID	FIRST_NAME	HIRE_DATE	Days
108	Mohammed	19-JAN-01	1
118	Sreenivas	29-JAN-01	3
116	Naresh	29-JAN-01	3
110	Regina	16-FEB-01	1
113	Chandan	10-MAR-01	1
126	Nataraj	14-JUN-01	1
115	Prasanth	16-JUL-01	1
106	Uedshree	17-MAR-04	1
107	Aarti	04-MAY-04	1
102	Stuti	26-JUL-04	1
101	Susmita	19-OCT-04	1
104	Harshada	30-NOV-04	1
103	Sahil	19-JUN-05	1

13 rows selected.

SELECT AVG(case when salary<20000 then 20000 else salary end)"Average" FROM Employee2;

```
SQL> SELECT AVG(case when salary<20000 then 20000 else salary end)"Average" FROM Employee2;
```

```
Average
-----
44538.4615
```

SELECT first_name,dept_id,AVG(case when salary>20000 then 20000 else 0 end)

OVER(partition by dept_id)"Average" FROM Employee2;

```
SQL> SELECT first_name,dept_id,AVG(case when salary>20000 then 20000 else 0 end)
2 OVER(partition by dept_id)"Average" FROM Employee2;
```

FIRST_NAME	DEPT_ID	Average
Susmita	10	16666.6667
Aarti	10	16666.6667
Uedshree	10	16666.6667
Stuti	10	16666.6667
Sahil	10	16666.6667
Harshada	10	16666.6667
Prasanth	20	10000
Naresh	20	10000
Sreenivas	29	0
Regina	30	20000
Chandan	30	20000
Nataraj	40	0
Mohammed	50	20000

13 rows selected.

PRACTICAL NO 3

Aim- Implementation of ORDBMS concepts like ADT (Abstract Data Types), Object table and Inheritance.

create or replace type ty_address as object

(street varchar(20),city varchar(20), pincode number(10));

```
SQL> create or replace type ty_address as object
2      (street varchar(20),city varchar(20), pincode number(10));
3      /
```

Type created.

create

or replace type ty_name as object

(fname varchar (20), mname varchar (20), lname varchar (20));

```
SQL> create or replace type ty_name as object
2      (fname varchar (20), mname varchar (20), lname varchar (20));
3      /
```

Type created.

Create table customer_adbt

(c_id number(5) primary key, c_name ty_name, c_add ty_address, c_phno number(10));

```
SQL> Create table customer_adbt
2      ( c_id number(5) primary key, c_name ty_name, c_add ty_address, c_phno number(10) );
```

Table created.

insert into customer_adbt values (2,ty_name('Susmita','B','Raul'),ty_address('Marine 2 Lines','Mumbai',400028),9820173925);

insert into customer_adbt values (1,ty_name('Harshada','S','Deshmukh'),ty_address('IIT 2 Powai','Mumbai',400076),9892750112);

insert into customer_adbt values

(3,ty_name('Sneha','M','Singh'),ty_address('Churhgate','Mumbai',400036),9899162616);

```
SQL> insert into customer_adbt values (2,ty_name('Susmita','B','Raul'),ty_address('Marine
2      2 Lines','Mumbai',400028),9820173925);
```

1 row created.

SQL>

```
SQL> insert into customer_adbt values (1,ty_name('Harshada','S','Deshmukh'),ty_address('IIT
2      2 Powai','Mumbai',400076),9892750112);
```

1 row created.

SQL> insert into customer_adbt values

```
2      (3,ty_name('Sneha','M','Singh'),ty_address('Churhgate','Mumbai',400036),9899162616);
```

1 row created.

select c.c_name.fname from customer_adbt c;

```
SQL> select c.c_name.fname from customer_adbt c;
```

C_NAME.FNAME

Susmita
Harshada
Sneha

select c.c_name.fname||' '||c.c_name.mname name from customer_adbt c;

```
SQL> select c.c_name.fname||' '||c.c_name.mname name from customer_adbt c;
```

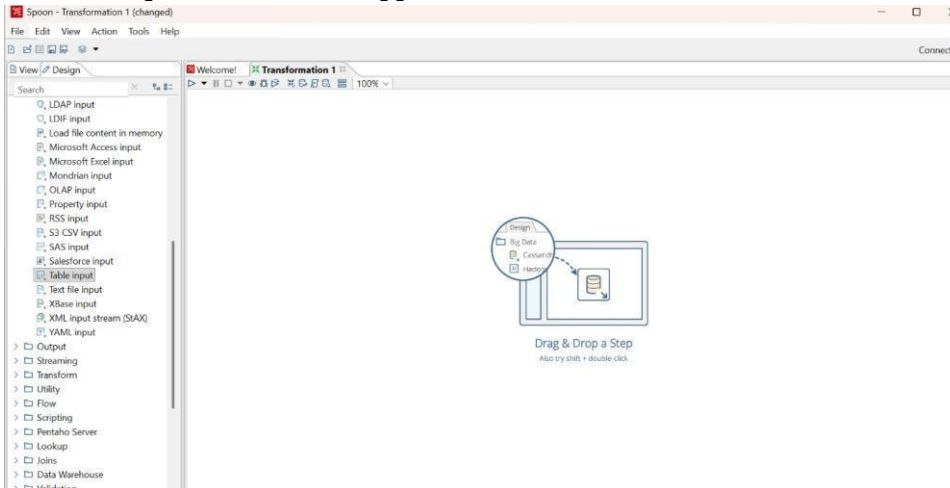
NAME

Susmita B
Harshada S
Sneha M

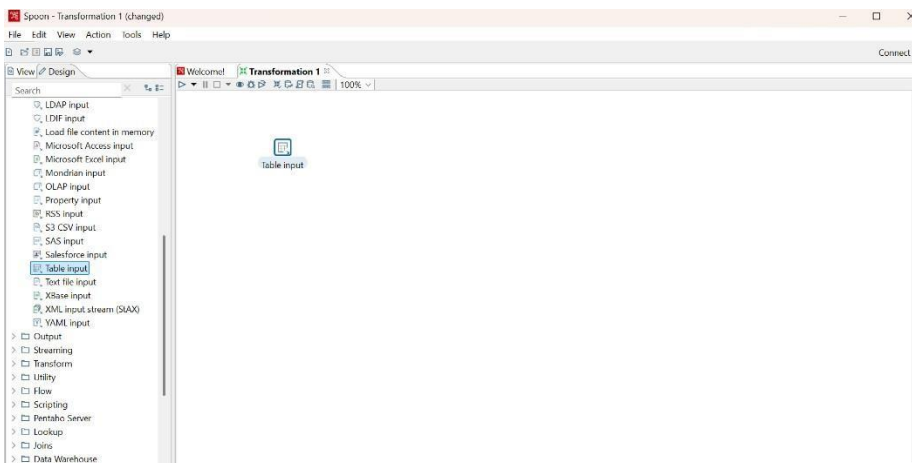
PRACTICAL NO 4

AIM-Implementation of ETL transformation with Pentaho like Copy data from Source (Table/Excel/ Oracle) and store it to Target (Table / Excel / Oracle), Adding sequence, Adding Calculator, Concatenation of two fields, splitting of two fields, Number Range, String Operations, Sorting data, Implement the merge join transformation on tables, Implement data validations on the table data.

STEP 1 – Open the Pentaho Application

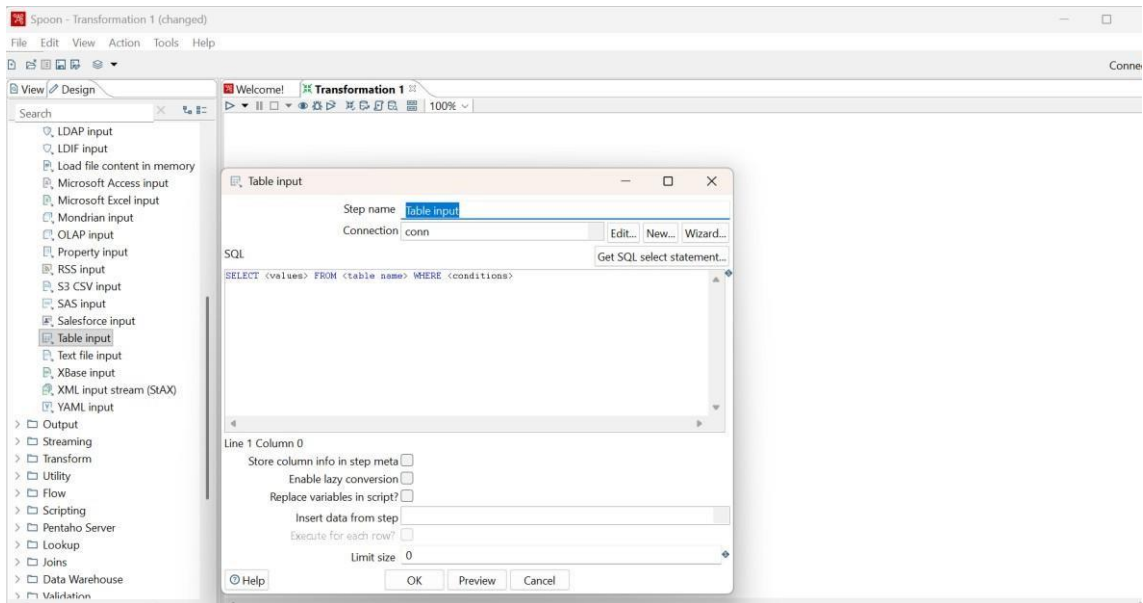


STEP 2- Click on the input option and take the Table input

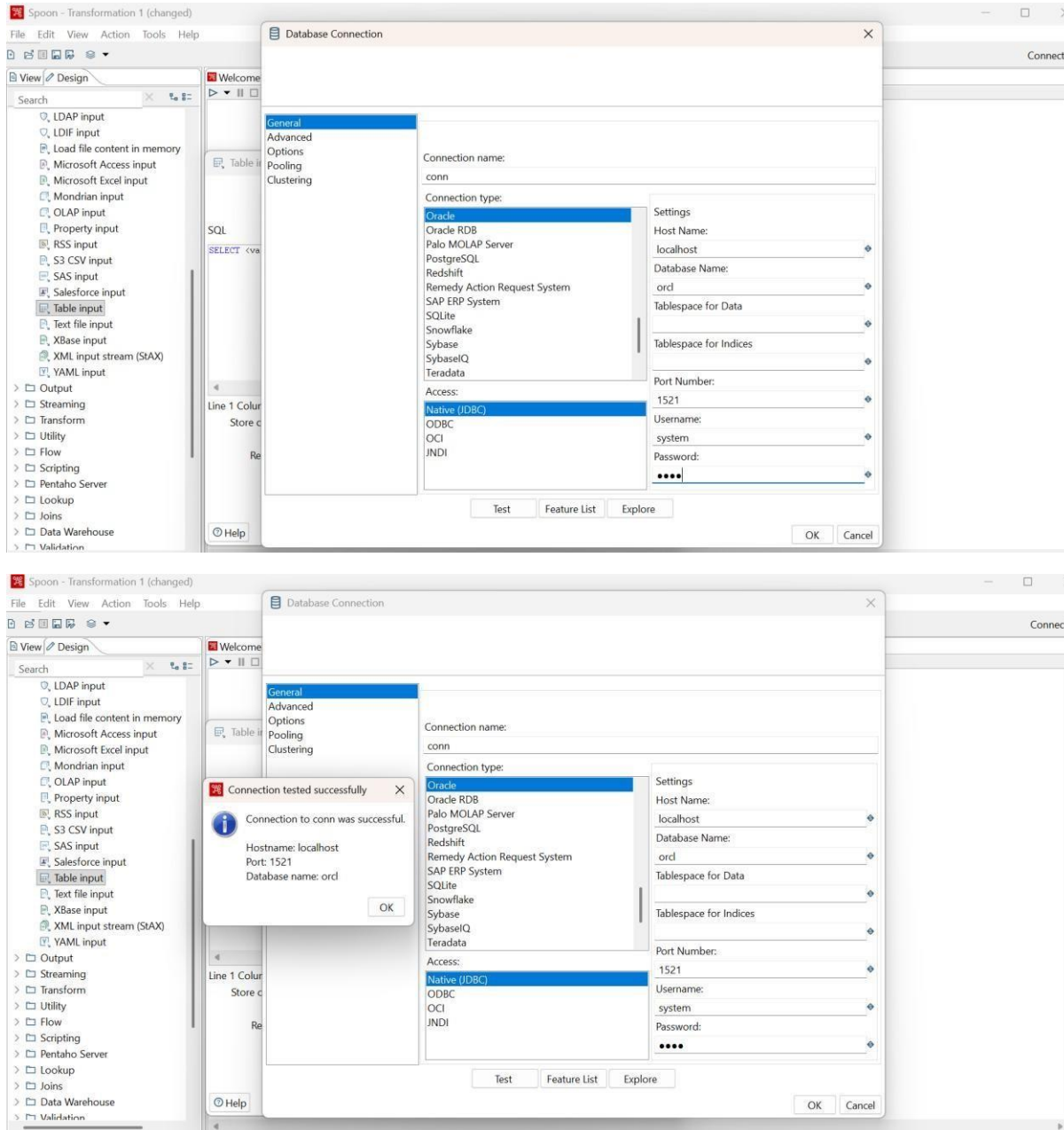


STEP 3- Click on the Table input and file the details like Connection name , Database name , Username and password

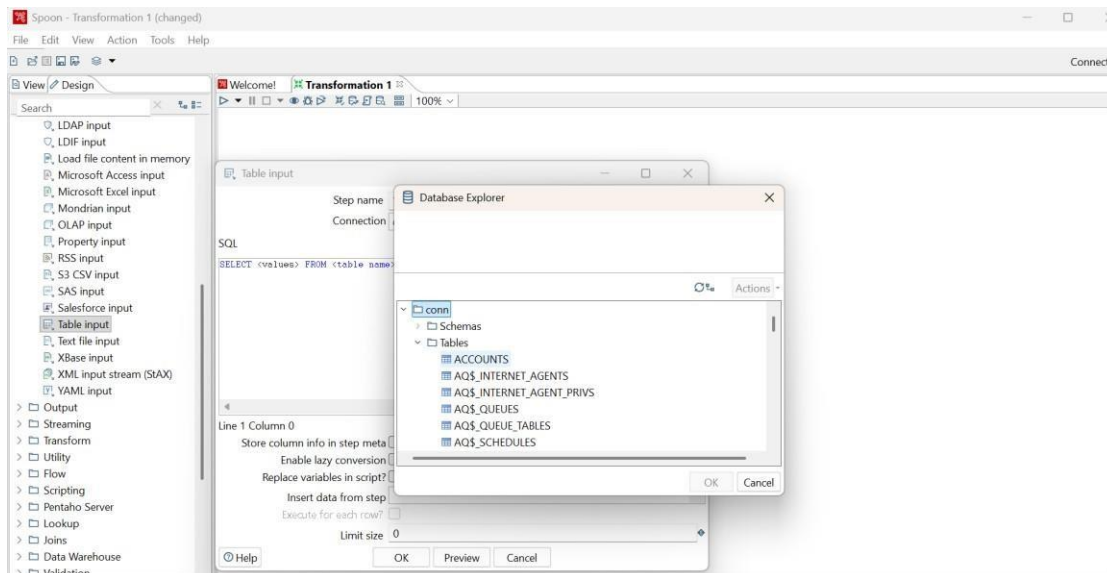
. Note username and password should be of the database which used



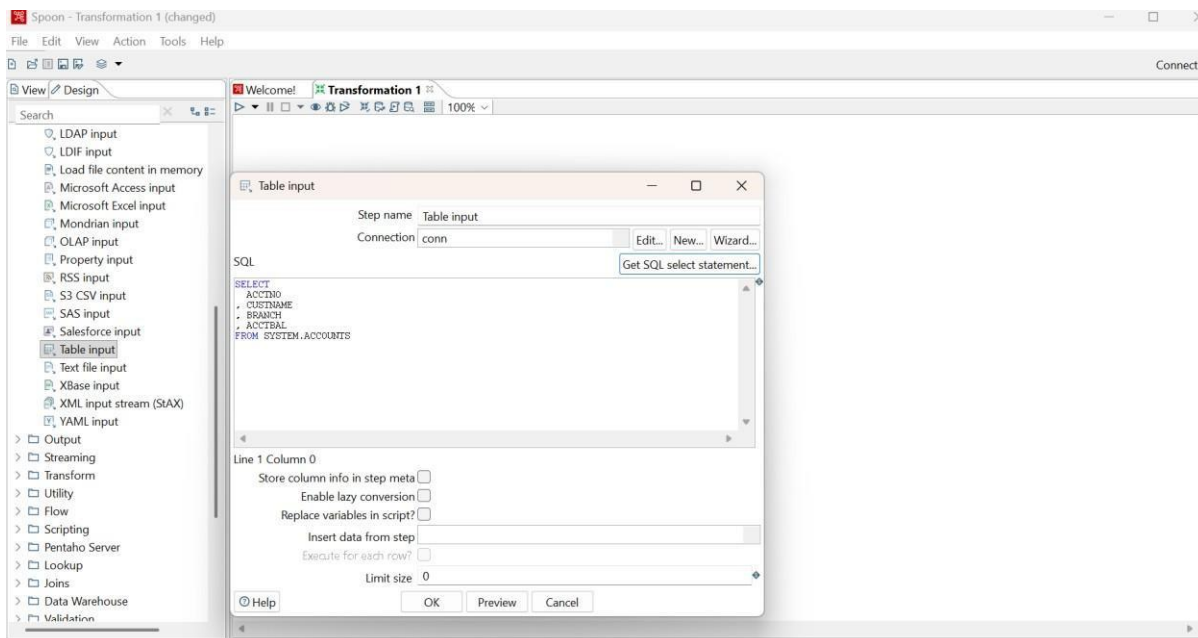
STEP 4 After all that click on Test and It will display connection test successfully



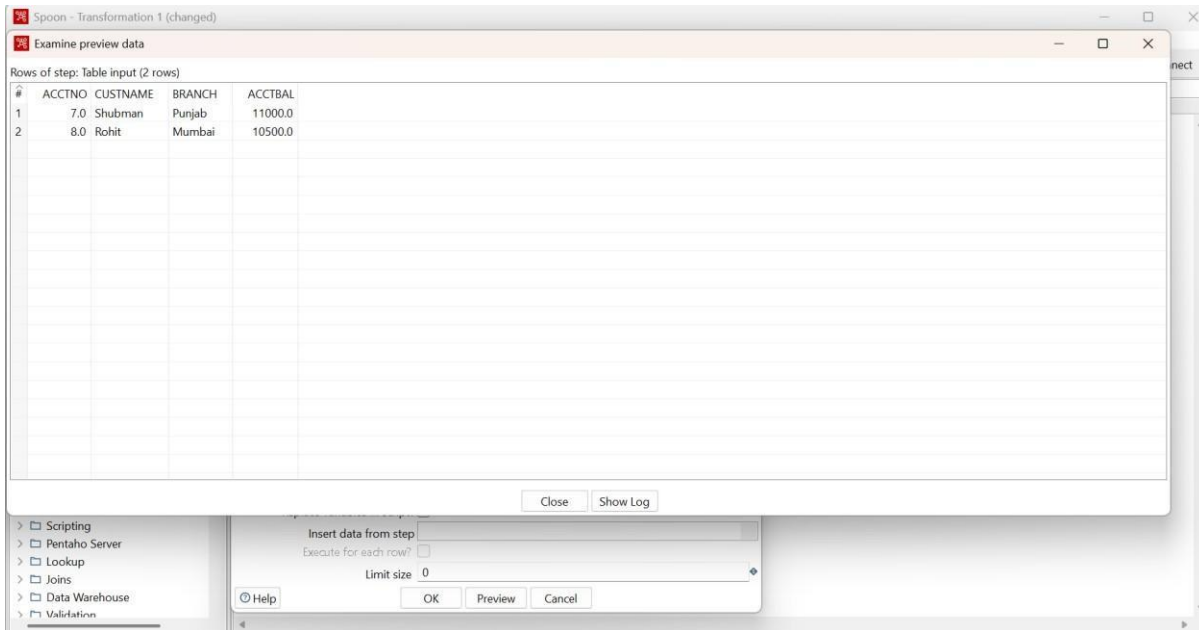
STEP 5- Click on the Get SQL Select statement and select the database



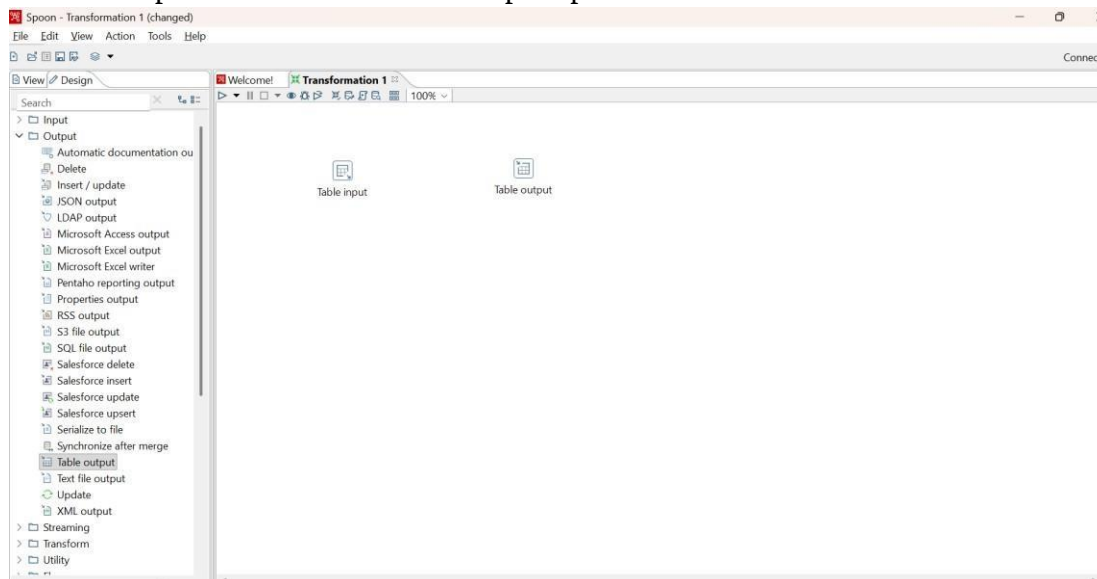
STEP 6 After that you will see the query



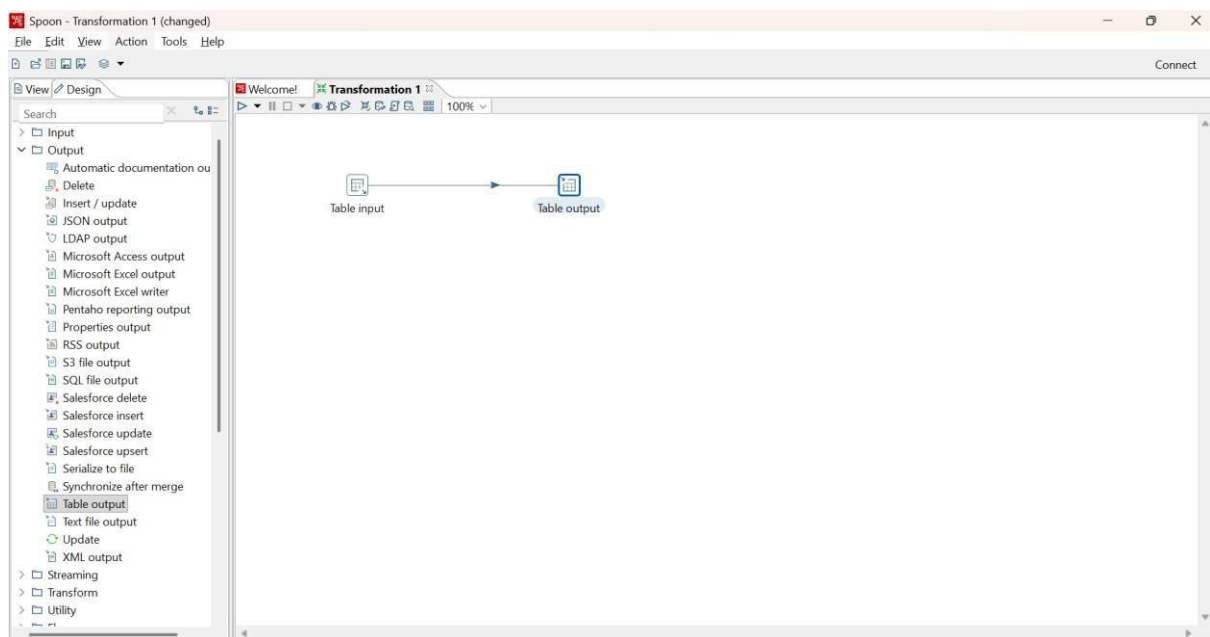
STEP 7 – Click on the preview to see the entries



STEP 8- From Output section select Table output option

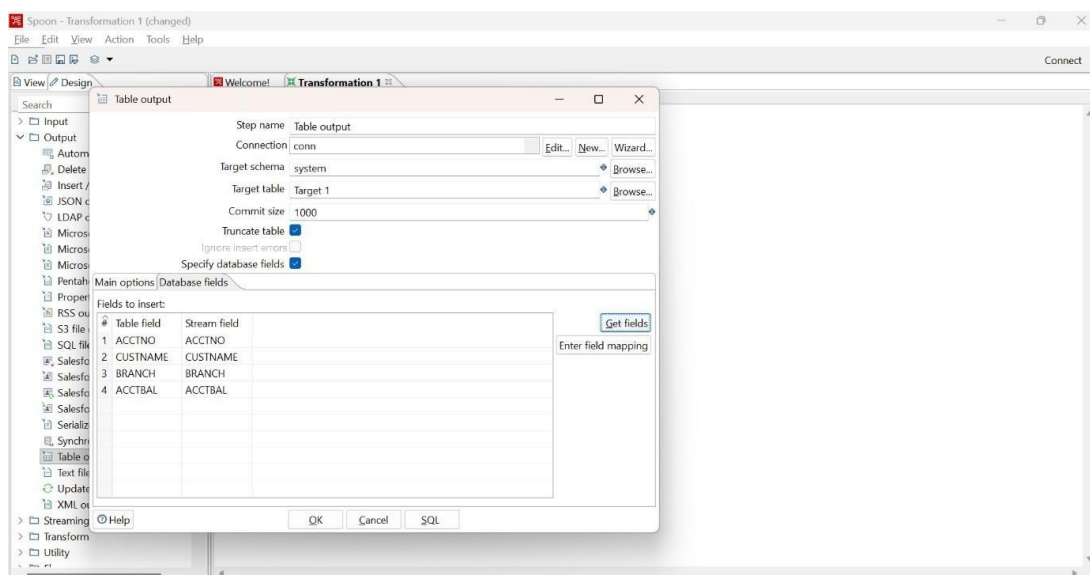


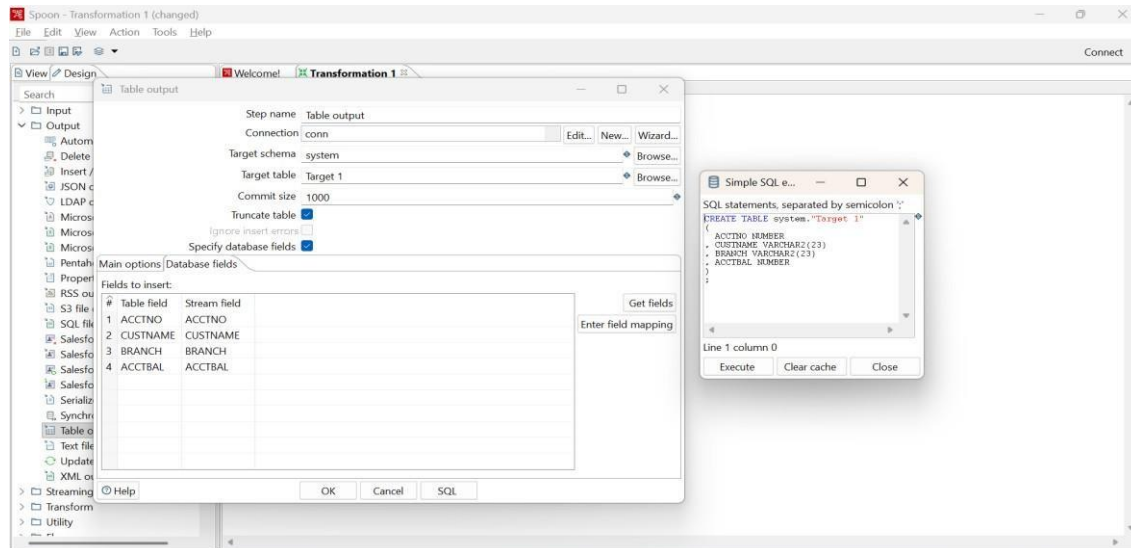
STEP 9 Drag an mouse pointer from table input to table table output

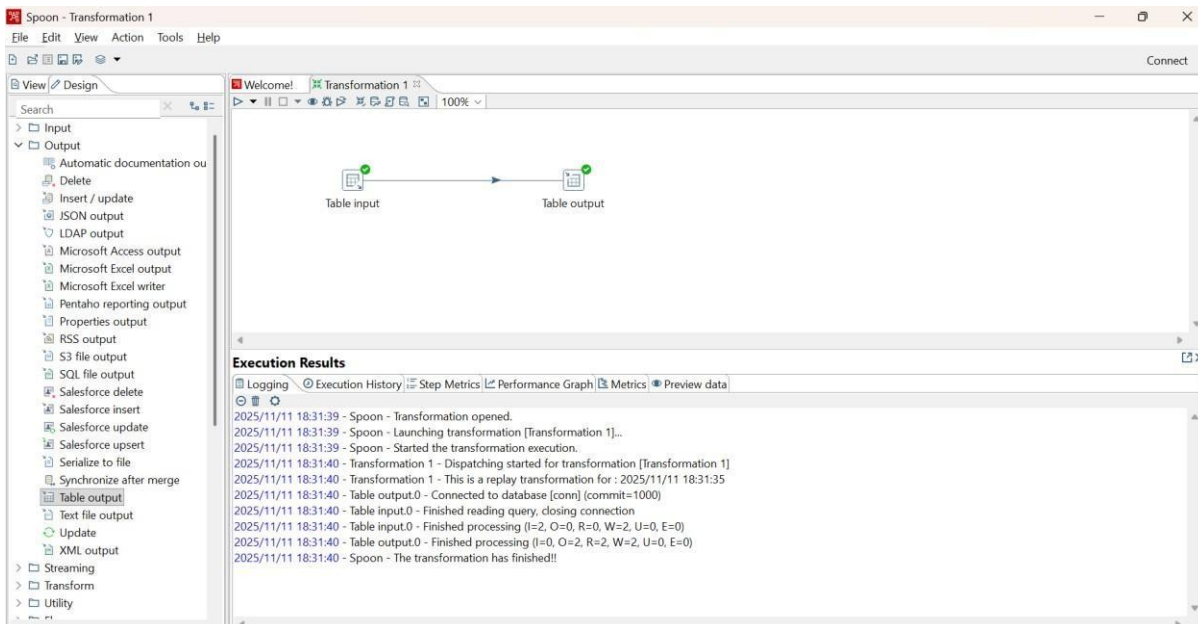
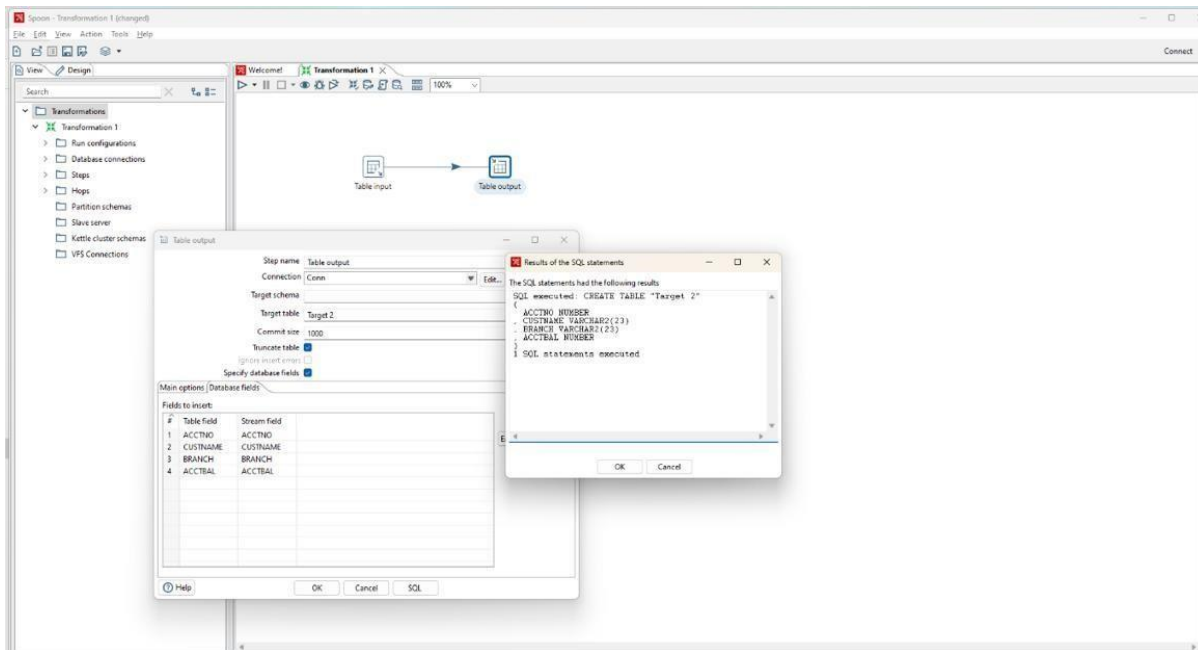


STEP 10- Click the Table output and file the Connection , Target Table, select the truncate table and Specify database

fields options

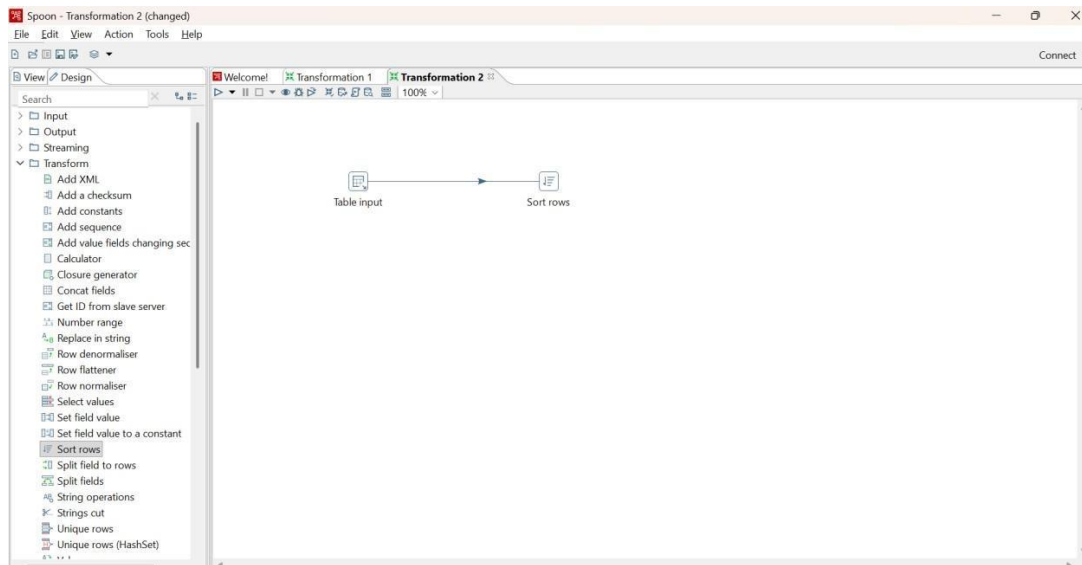




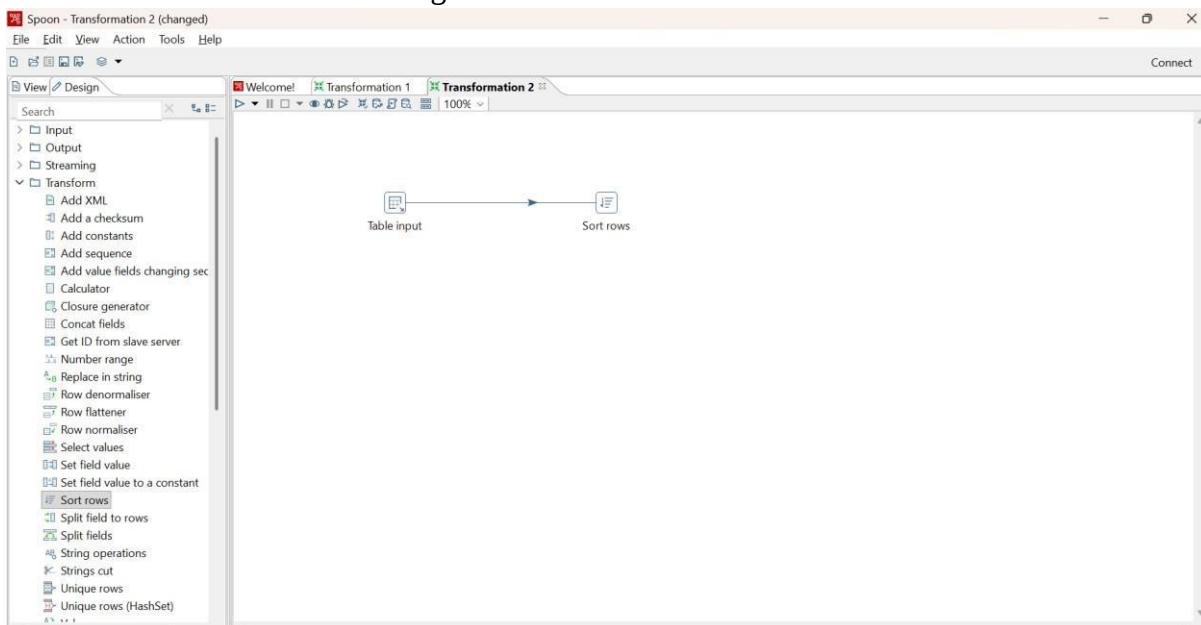


2. Sorting Operation and adding Sequence to Output Table.

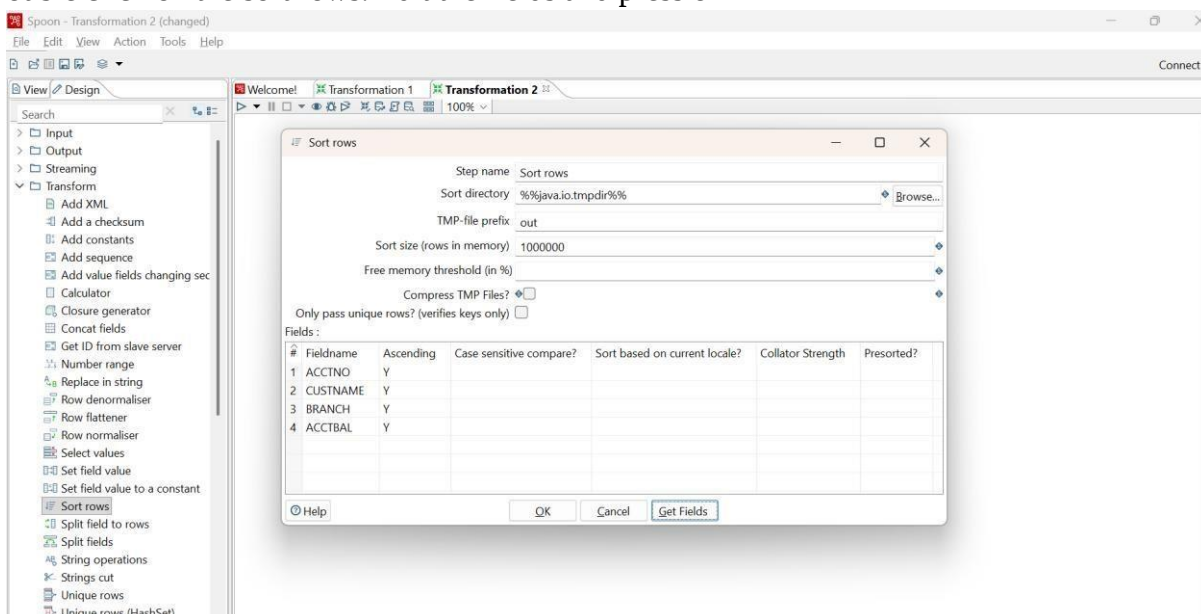
Perform the first 5 steps same as 1.



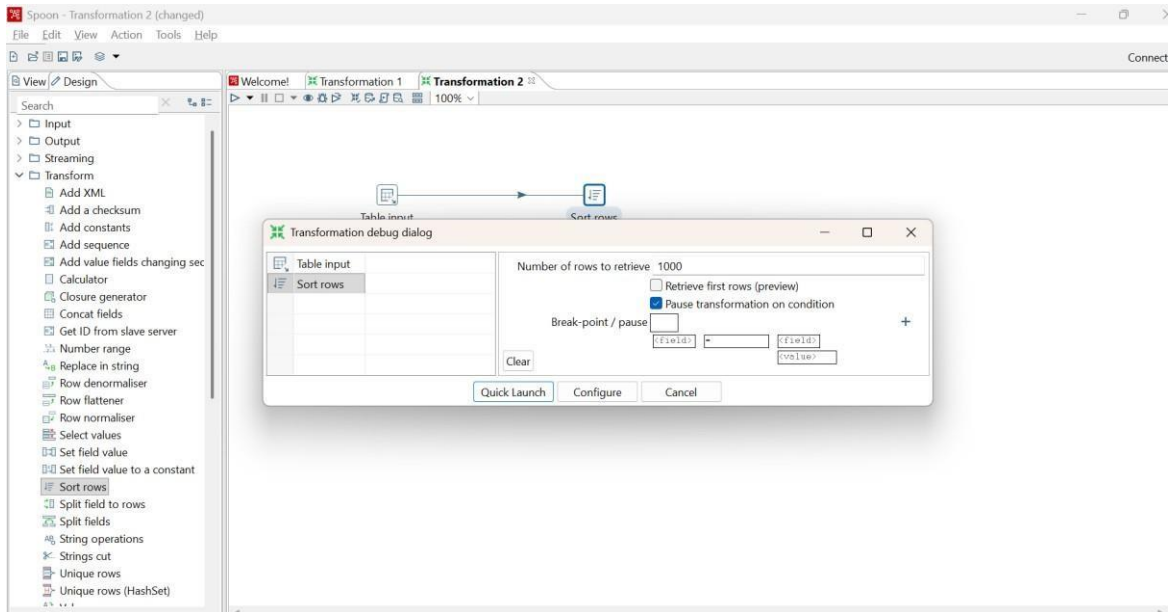
Click on Transform in the left. Drag sort rows on the Panel



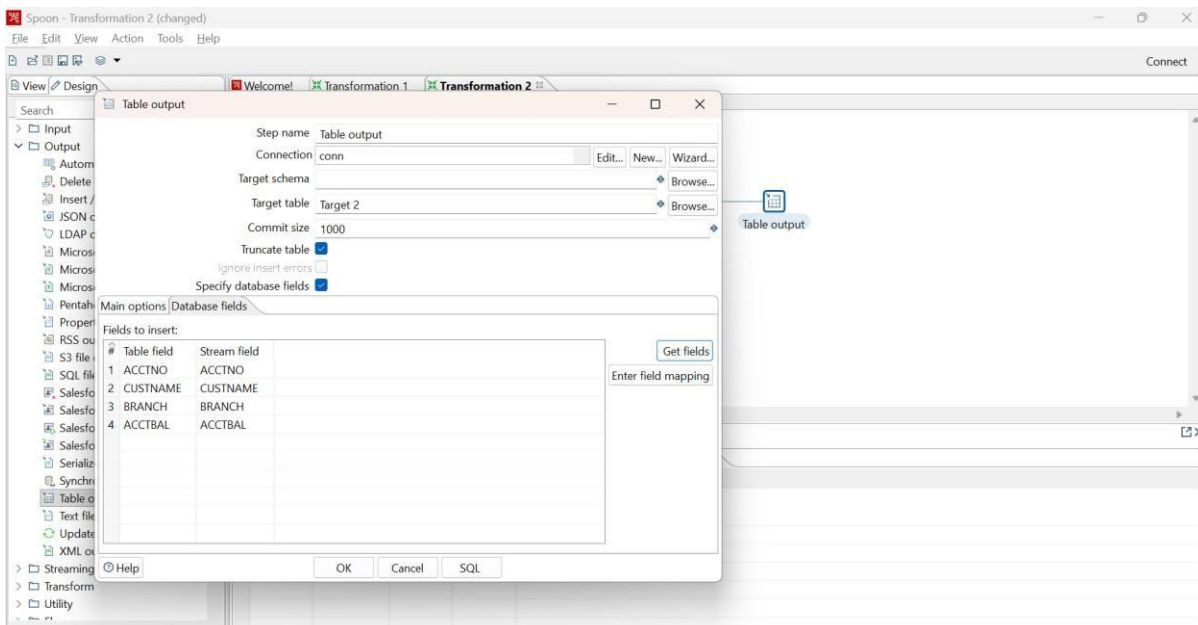
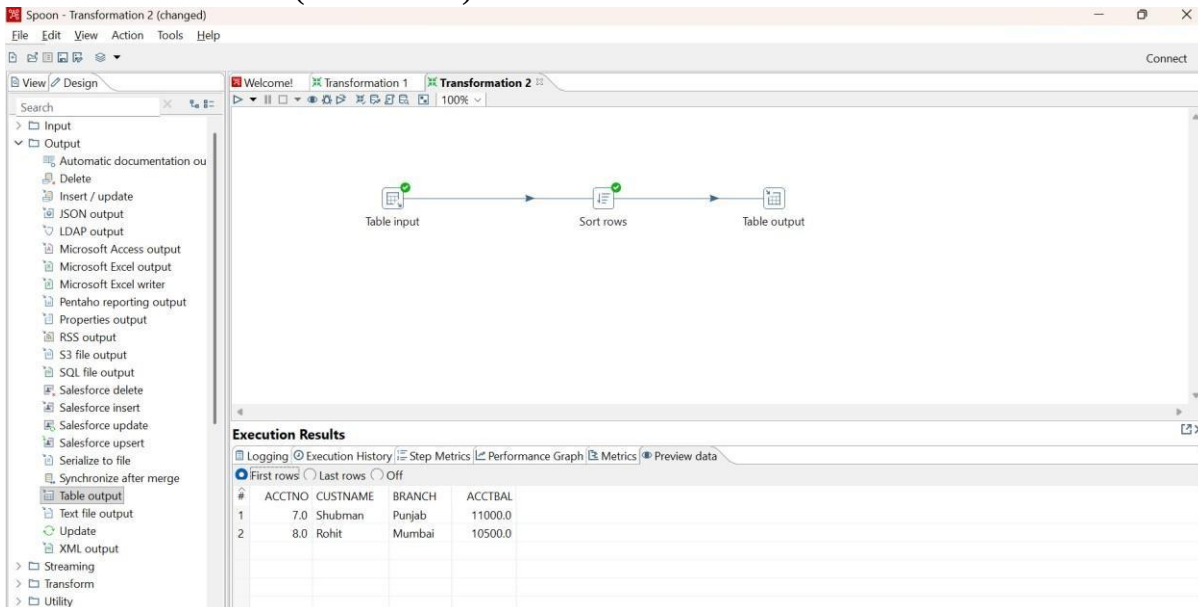
Double click on the sort rows. Edit the fields and press ok



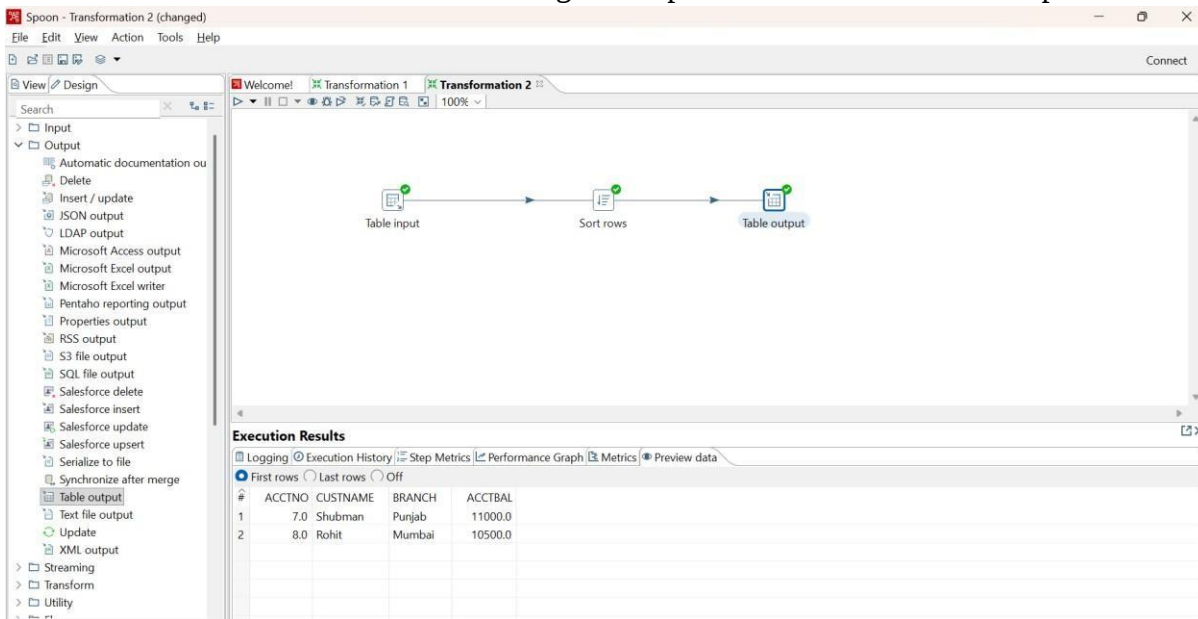
Click on Debug the Transformation and Click on Quick Launch.



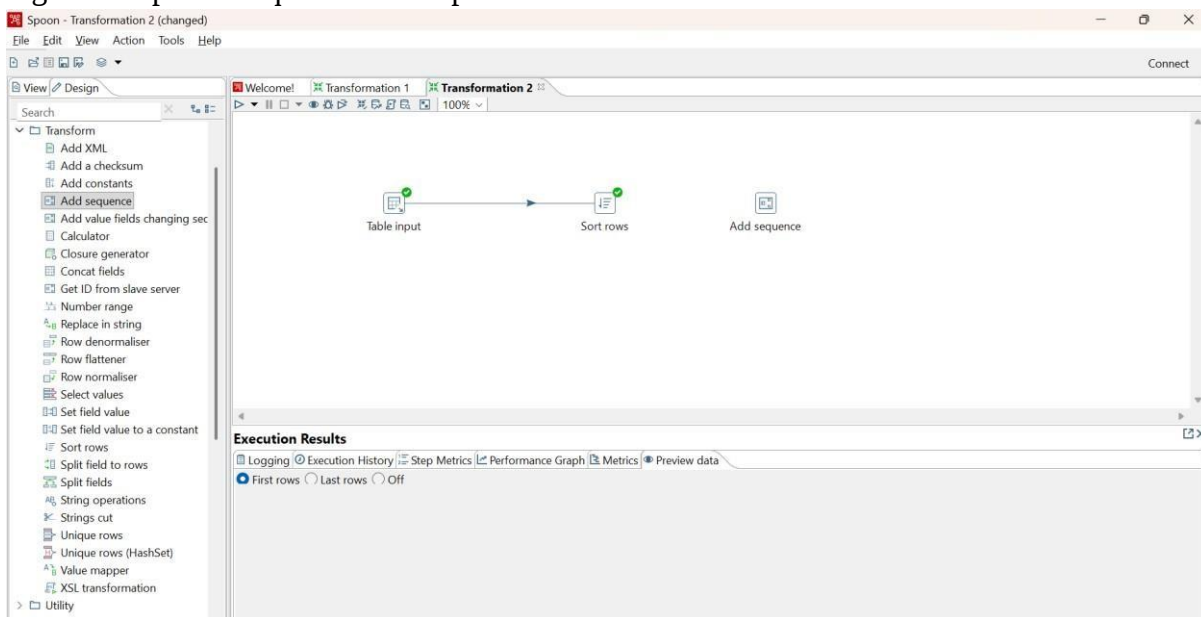
Successful connections(Green Ticks)



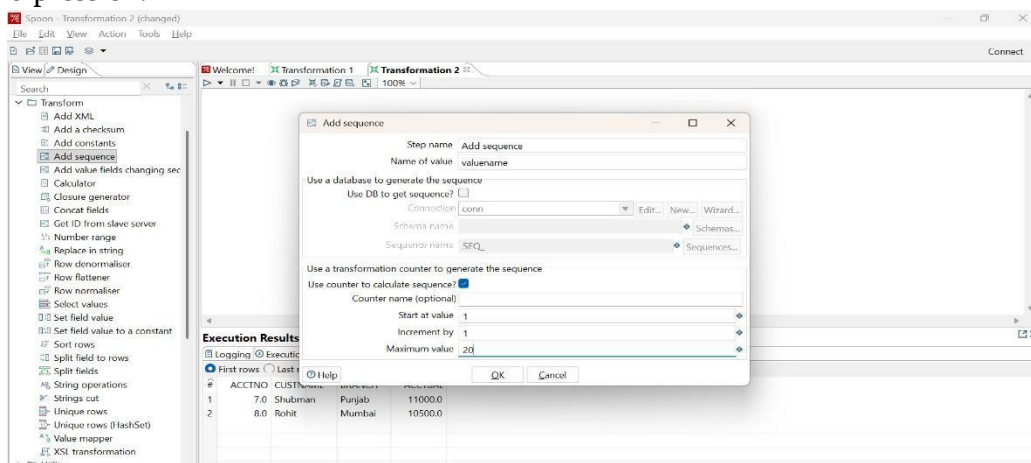
hold the cursor on the sort row and then drag the output connector to the table output.



Drag and drop Add sequence on the panel



Double Click on Add Sequence Control. Enter the values for Start at Values, Increment by and Maximum Values and press ok.



Click on Debug the Transformation and click on Quick Launch

Spoon - Transformation 2 (changed)

File Edit View Action Tools Help

View Design

Search

Transform

- Add XML
- Add a checksum
- Add constants
- Add sequence
- Add value fields changing sec
- Calculator
- Closure generator
- Concat fields
- Get ID from slave server
- Number range
- Replace in string
- Row denormaliser
- Row flattener
- Row normaliser
- Select values
- Set field value
- Set field value to a constant
- Sort rows
- Split field to rows
- Split fields
- String operations
- Strings cut
- Unique rows
- Unique rows (HashSet)
- Value mapper
- XSL transformation

Utility

Transformation debug dialog

Table input

Sort rows

Add sequence

Number of rows to retrieve: 1000

☐ Retrieve first rows (preview)

☒ Pause transformation on condition

Break-point / pause

Clear

Quick Launch Configure Cancel

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

First rows Last rows Off

#	ACCTNO	CUSTNAME	BRANCH	ACCTBAL
1	7.0	Shubman	Punjab	11000.0
2	8.0	Rohit	Mumbai	10500.0

Spoon - Transformation 2 (changed)

File Edit View Action Tools Help

View Design

Search

Transform

- Add XML
- Add a checksum
- Add constants
- Add sequence
- Add value fields changing sec
- Calculator
- Closure generator
- Concat fields
- Get ID from slave server
- Number range
- Replace in string
- Row denormaliser
- Row flattener
- Row normaliser
- Select values
- Set field value
- Set field value to a constant
- Sort rows
- Split field to rows
- Split fields
- String operations
- Strings cut
- Unique rows
- Unique rows (HashSet)
- Value mapper
- XSL transformation

Utility

Transformation debug dialog

Table input

Sort rows

Add sequence

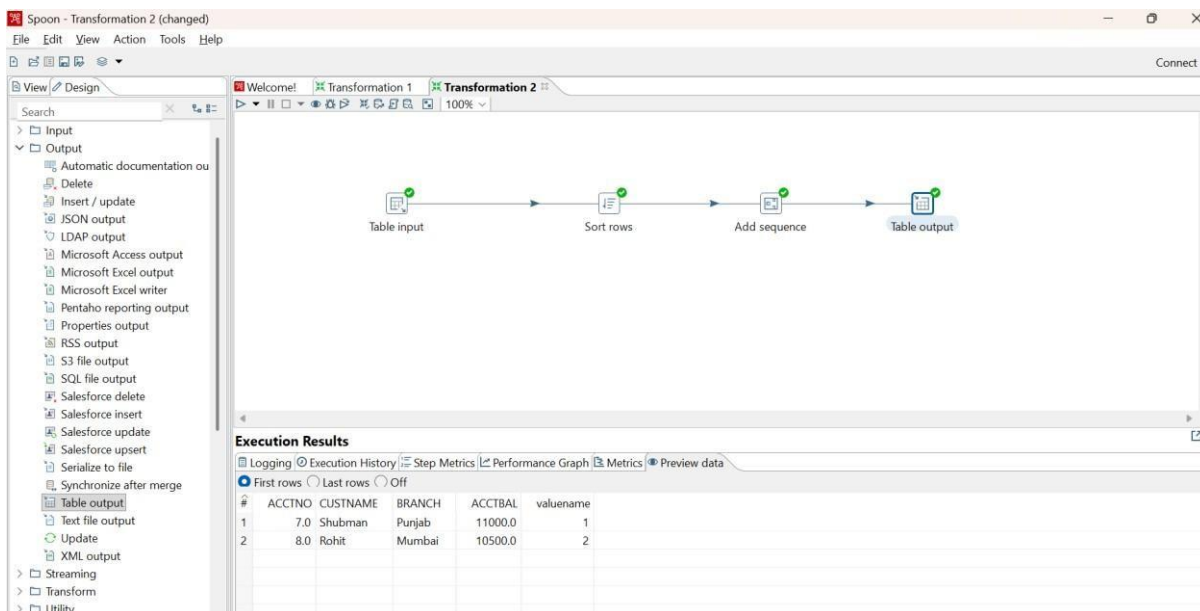
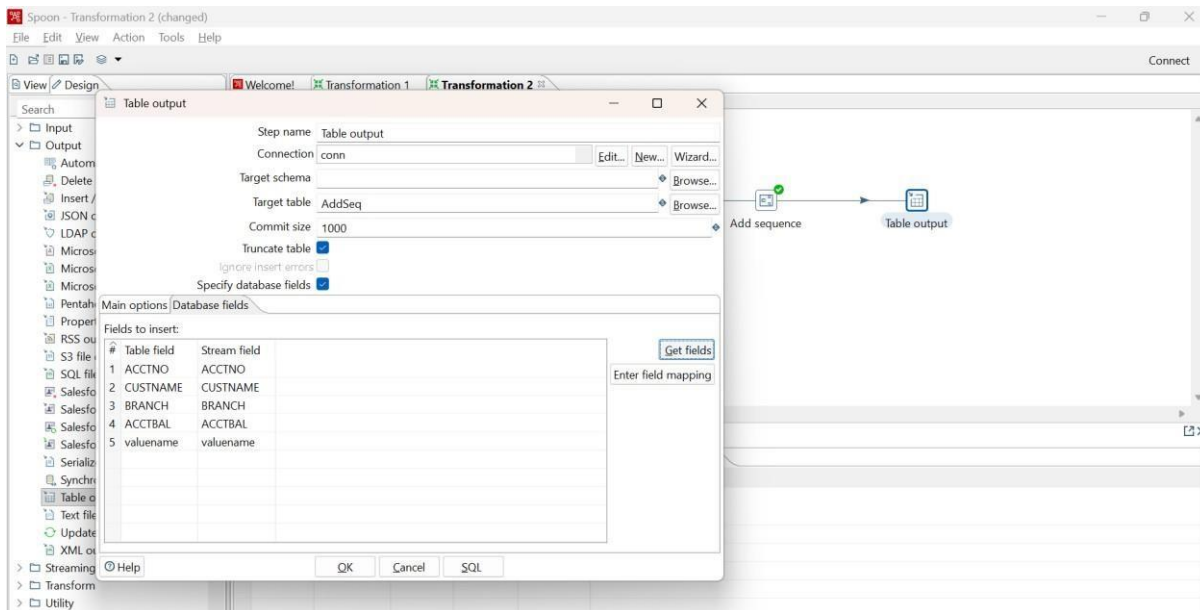
Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

First rows Last rows Off

#	ACCTNO	CUSTNAME	BRANCH	ACCTBAL	valuenam
1	7.0	Shubman	Punjab	11000.0	1
2	8.0	Rohit	Mumbai	10500.0	2

Drag and drop the table output on the panel. Double Click on the table output

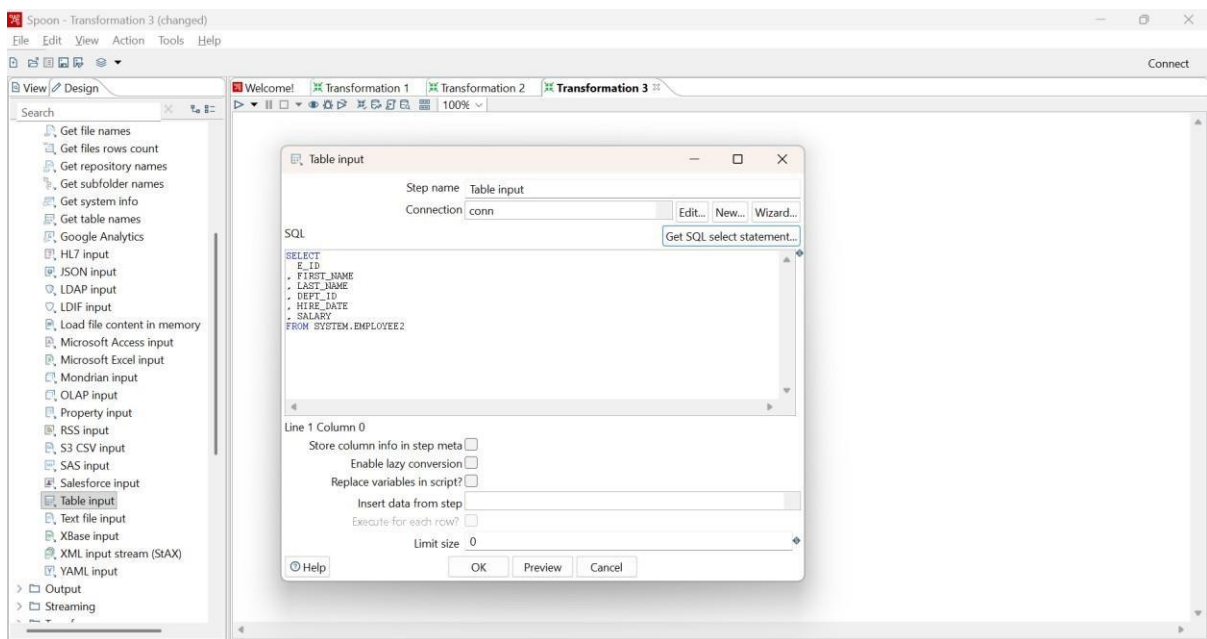
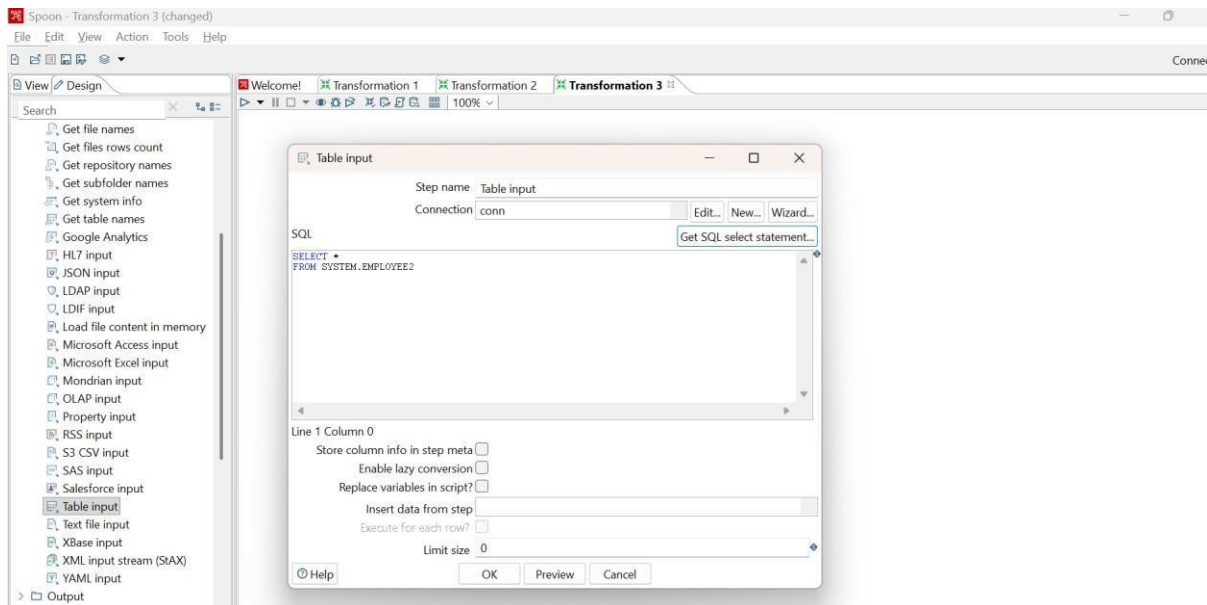


SQL> select * from AddSeq;

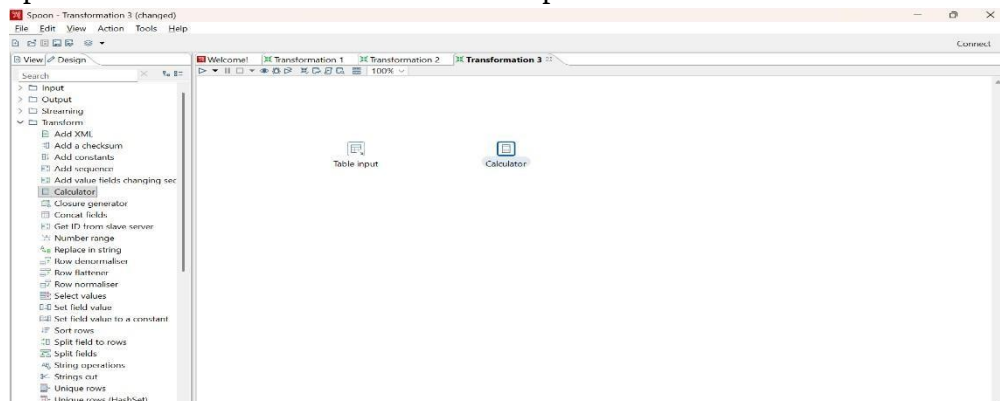
ACCTNO	CUSTNAME	BRANCH	ACCTBAL	VALUENAME
7	Shubman	Punjab	11000	1
8	Rohit	Mumbai	10500	2

3. Calculator Operation

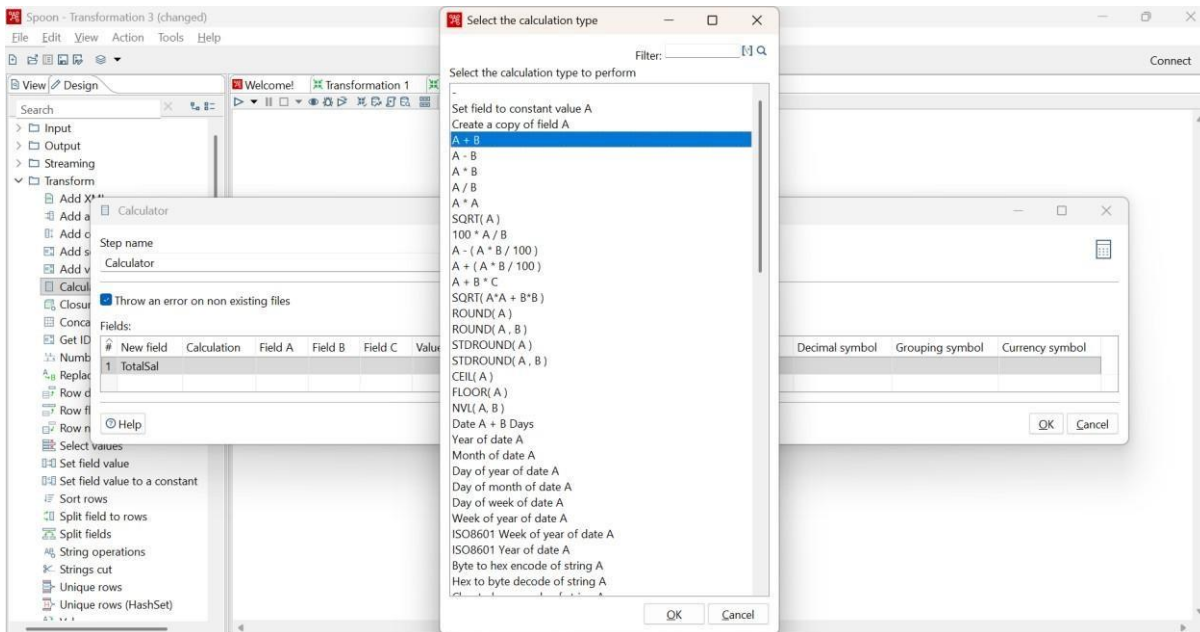
Step 1: Perform first 5 steps same as practical 1.



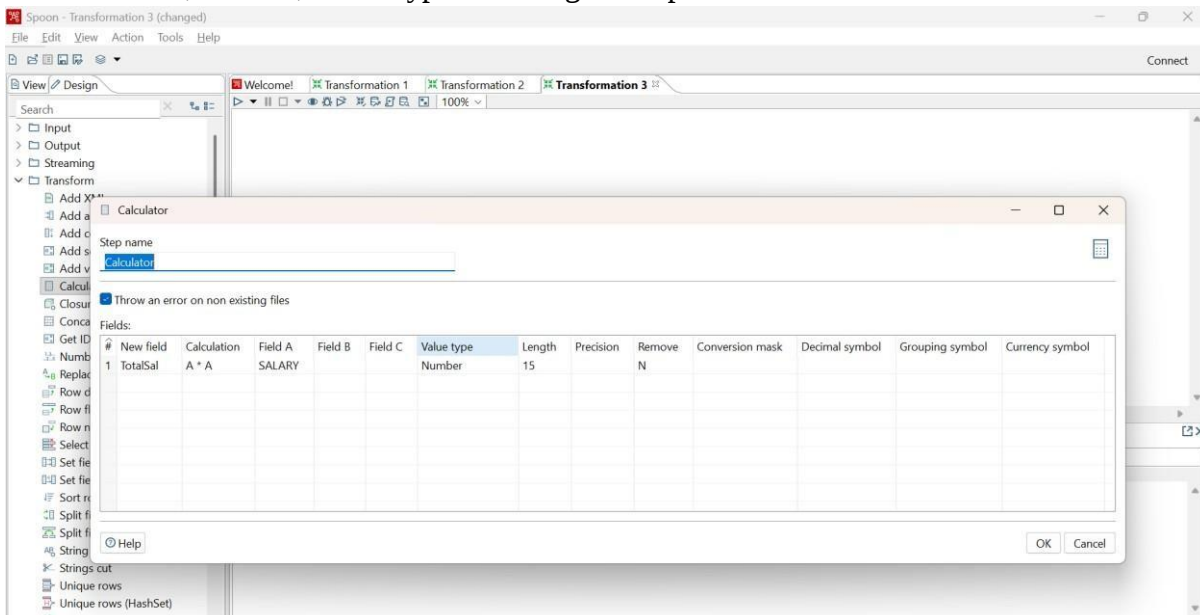
Drag and Drop Calculator from transformation on the panel.



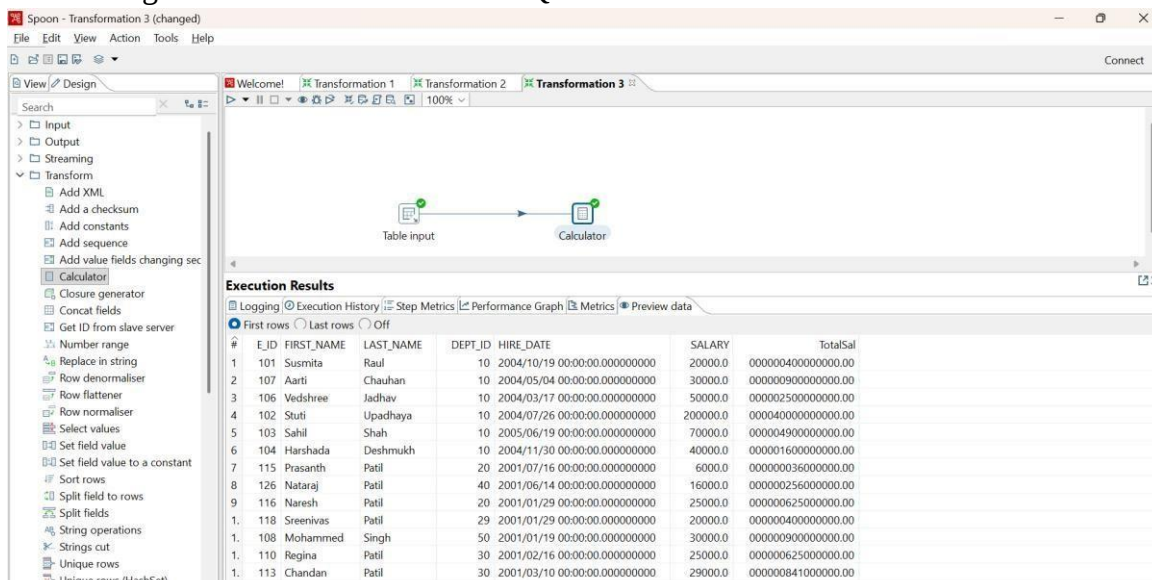
Double Click on the Calculator a.
specify the new field name
b. select the calculation function

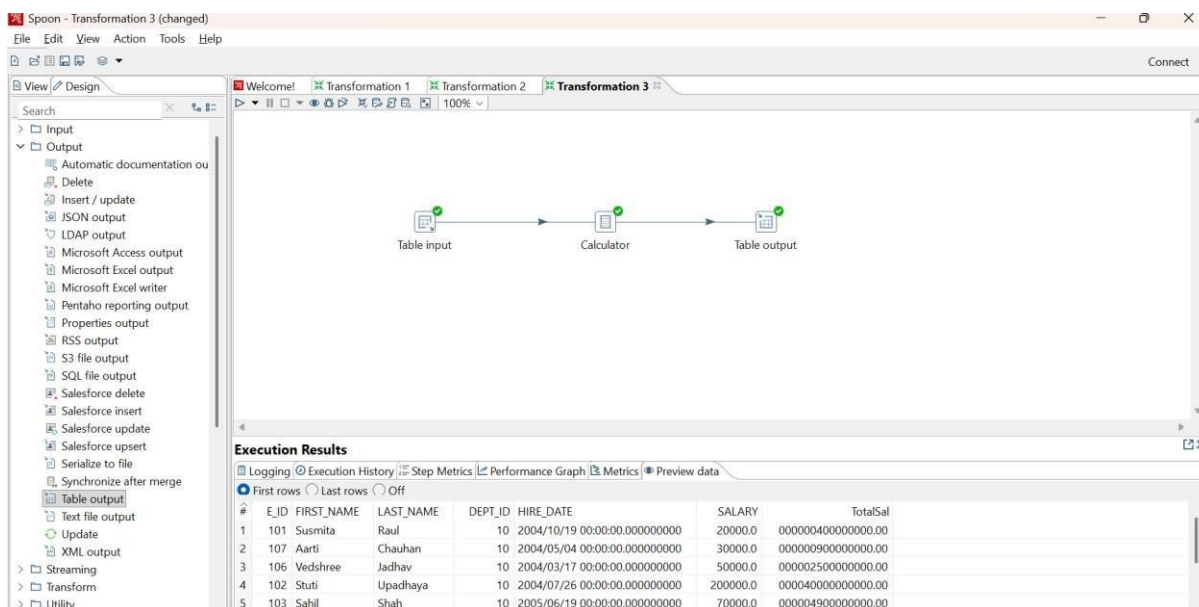
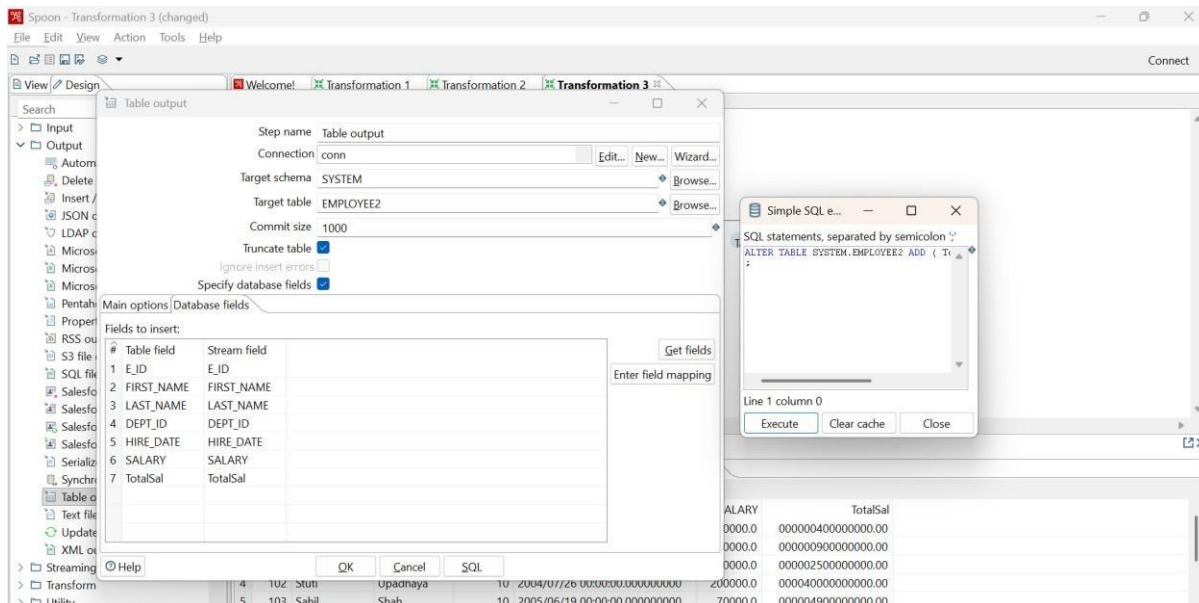


c. Enter Field A, Field B, Value type and Length and press ok.



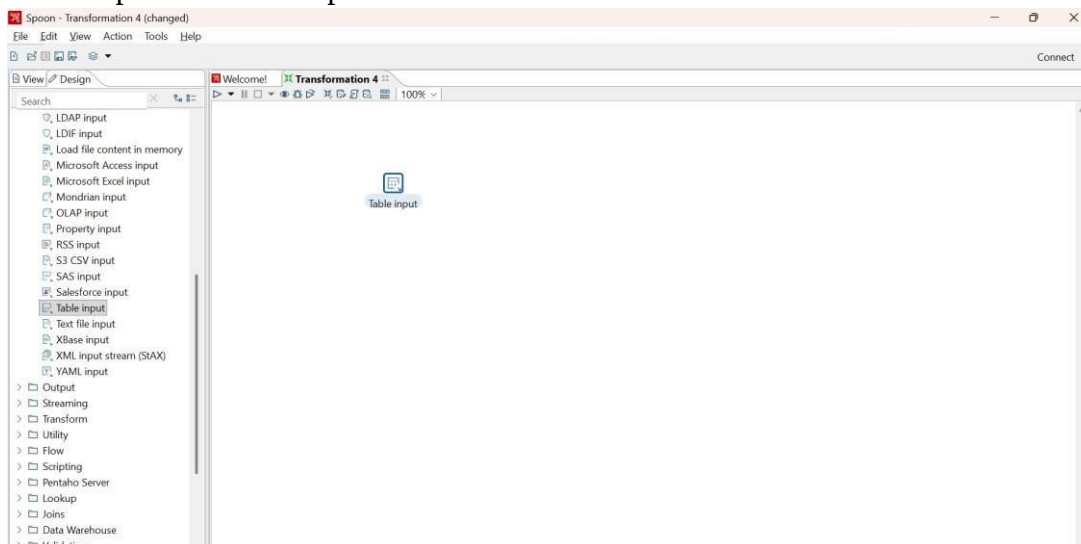
Click on Debug transformation and click on Quick Launch.



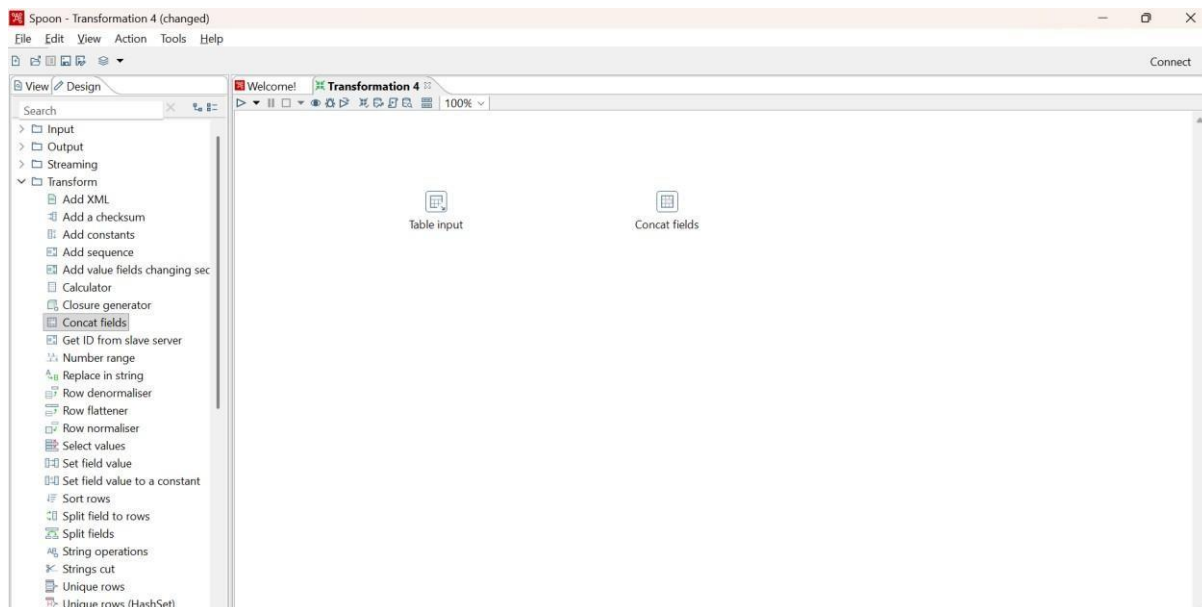


4. Concatenation Operation

Perform the steps 6-10 same as practical 1.

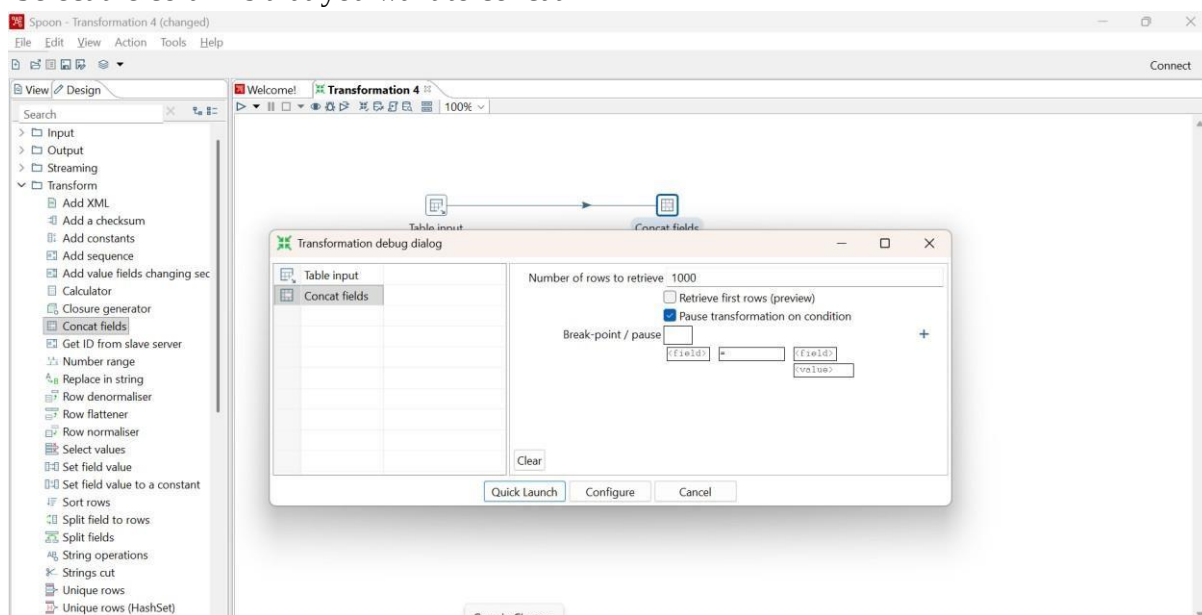


Drag and drop Concat Field



Double Click on Concat Field

- Enter the values for Target Field Name, Length of the Target Field and the Separator and press on Get Fields
- Select the columns that you want to concat



Click on
Debug

transformation and click on Quick Launch

Spoon - Transformation 4 (changed)

File Edit View Action Tools Help

View Design

Search

- Input
- Output
- Streaming
- Transform
 - Add XML
 - Add a checksum
 - Add constants
 - Add sequence
 - Add value fields changing sec
 - Calculator
 - Closure generator
 - Concat fields
 - Get ID from slave server
 - Number range
 - Replace in string
 - Row denormaliser
 - Row flattener
 - Row normaliser
 - Select values
 - Set field value
 - Set field value to a constant
 - Sort rows
 - Split field to rows
 - Split fields
 - String operations
 - Strings cut
 - Unique rows
 - Unique rows (HashSet)

Table input → Concat fields

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

First rows Last rows Off

#	E_ID	FIRST_NAME	LAST_NAME	DEPT_ID	HIRE_DATE	SALARY	TOTALSAL	FULLNAME
1	101	Susmita	Raul	10	2004/10/19 00:00:00.000000000	20000.0	400000000	101-Susmita -Raul
2	107	Aarti	Chauhan	10	2004/05/04 00:00:00.000000000	30000.0	900000000	107-Aarti -Chauhan
3	106	Vedshree	Jadhav	10	2004/03/17 00:00:00.000000000	50000.0	2500000000	106-Vedshree -Jadhav
4	102	Stuti	Upadhaya	10	2004/07/26 00:00:00.000000000	200000.0	4000000000	102-Stuti -Upadhaya
5	103	Sahil	Shah	10	2005/06/19 00:00:00.000000000	70000.0	4900000000	103-Sahil -Shah
6	104	Harshada	Deshmukh	10	2004/11/30 00:00:00.000000000	40000.0	1600000000	104-Harshada -Deshmukh
7	115	Prasanth	Patil	20	2001/07/16 00:00:00.000000000	6000.0	360000000	115-Prasanth -Patil
8	126	Nataraj	Patil	40	2001/06/14 00:00:00.000000000	16000.0	2560000000	126-Nataraj -Patil
9	116	Nareesh	Patil	20	2001/01/29 00:00:00.000000000	25000.0	6250000000	116-Nareesh -Patil
1.	118	Sreenivas	Patil	29	2001/01/29 00:00:00.000000000	20000.0	4000000000	118-Sreenivas -Patil
1.	108	Mohammed	Singh	50	2001/01/19 00:00:00.000000000	30000.0	9000000000	108-Mohammed -Singh
1.	110	Regina	Patil	30	2001/02/16 00:00:00.000000000	25000.0	6250000000	110-Regina -Patil

Preview

Spoon - Transformation 4 (changed)

File Edit View Action Tools Help

View Design

Search

- Input
- Output
- Streaming
- Transform
 - Add XML
 - Add a checksum
 - Add constants
 - Add sequence
 - Add value fields changing sec
 - Calculator
 - Closure generator
 - Concat fields
 - Get ID from slave server
 - Number range
 - Replace in string
 - Row denormaliser
 - Row flattener
 - Row normaliser
 - Select values
 - Set field value
 - Set field value to a constant
 - Sort rows
 - Split field to rows
 - Split fields
 - String operations
 - Strings cut
 - Unique rows
 - Unique rows (HashSet)

Concat fields

Step name: Concat fields

Target Field Name: FULLNAME

Length of Target Field: 20

Separator: -

Enclosure: *

Fields Advanced

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim Type	Nul
1	E_ID	Integer		4	0				none	
2	FIRST_NAME	String		15					none	
3	LAST_NAME	String		15					none	
4	DEPT_ID	Integer		3	0				none	
5	HIRE_DATE	Timestamp		0					none	
6	SALARY	Number	00000000.00	10	2				none	
7	TOTALSAL	Integer		15	0				none	

Get Fields Minimal width

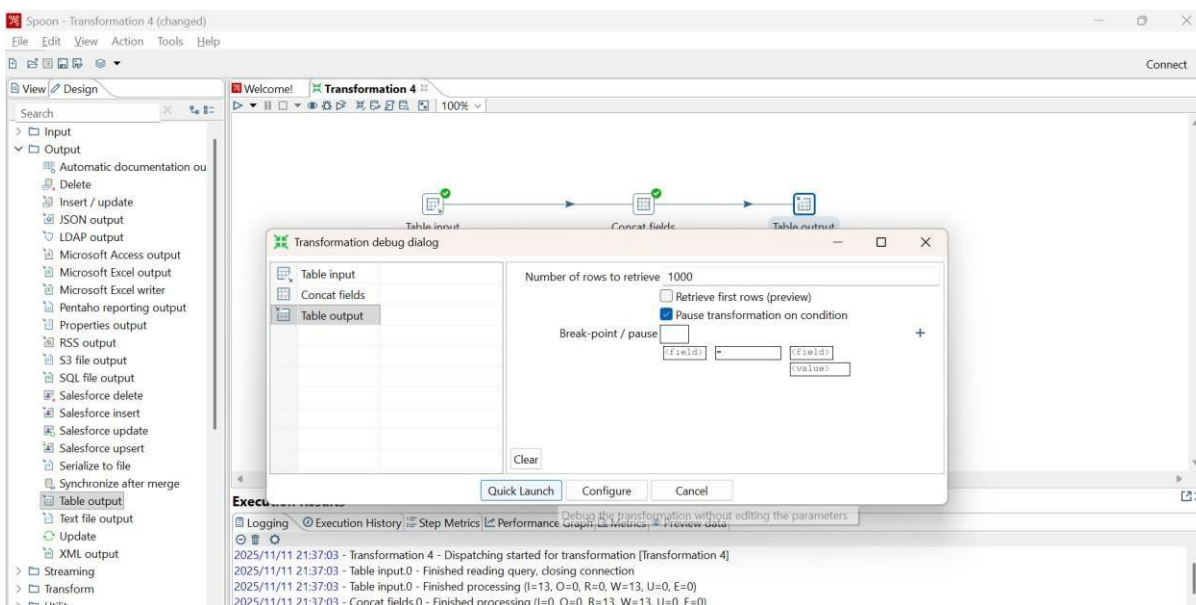
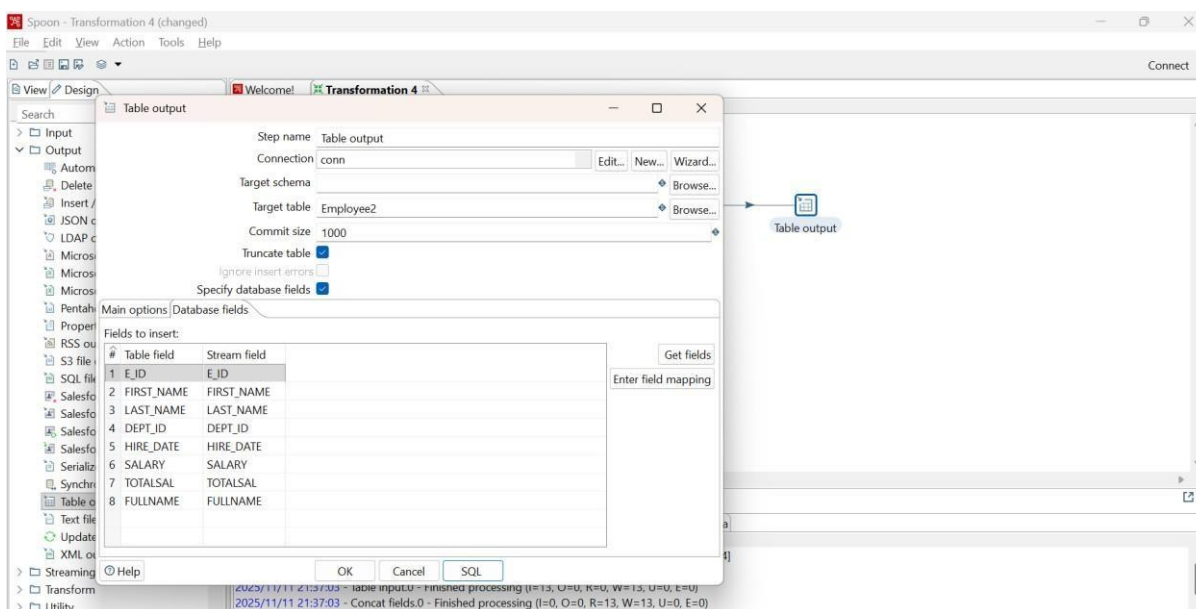
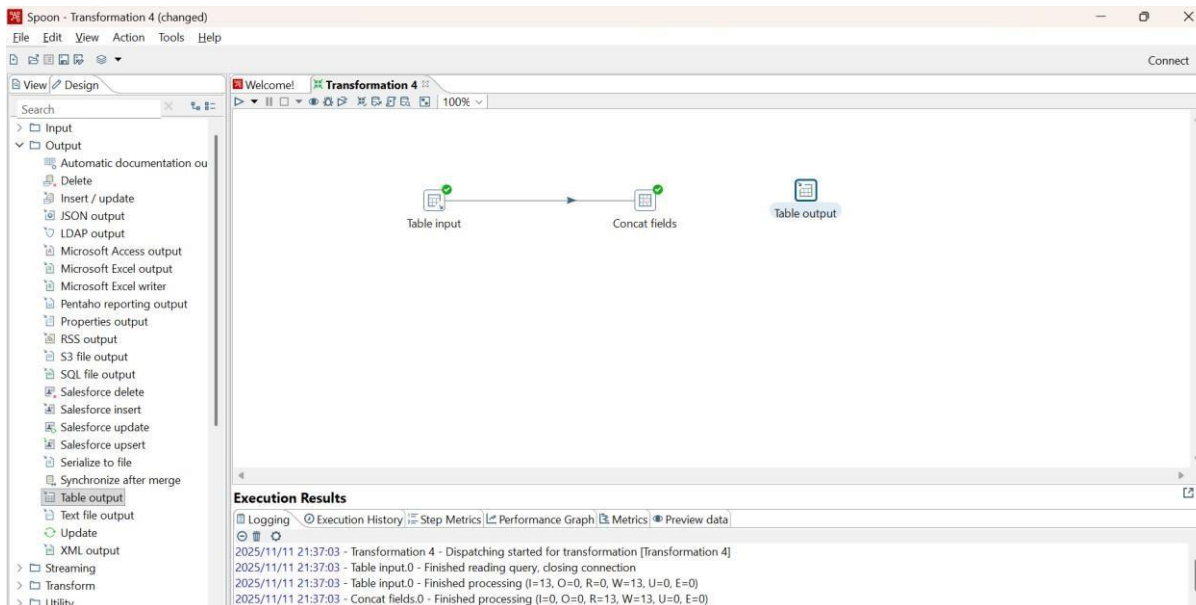
OK Cancel

Examine preview data

Rows of step: Concat fields (13 rows)

#	E_ID	FIRST_NAME	LAST_NAME	DEPT_ID	HIRE_DATE	SALARY	TOTALSAL	FULLNAME
1	113	Chandan	Patil	30	2001/03/10 00:00:00.000000000	29000.0	841000000	113-Chandan -Patil
2	110	Regina	Patil	30	2001/02/16 00:00:00.000000000	25000.0	6250000000	110-Regina -Patil
3	108	Mohammed	Singh	50	2001/01/19 00:00:00.000000000	30000.0	9000000000	108-Mohammed -Singh
4	118	Sreenivas	Patil	29	2001/01/29 00:00:00.000000000	20000.0	4000000000	118-Sreenivas -Patil
5	116	Nareesh	Patil	20	2001/01/29 00:00:00.000000000	25000.0	6250000000	116-Nareesh -Patil
6	126	Nataraj	Patil	40	2001/06/14 00:00:00.000000000	16000.0	2560000000	126-Nataraj -Patil
7	115	Prasanth	Patil	20	2001/07/16 00:00:00.000000000	6000.0	360000000	115-Prasanth -Patil
8	104	Harshada	Deshmukh	10	2004/11/30 00:00:00.000000000	40000.0	1600000000	104-Harshada -Deshmukh
9	103	Sahil	Shah	10	2005/06/19 00:00:00.000000000	70000.0	4900000000	103-Sahil -Shah
1.	102	Stuti	Upadhaya	10	2004/07/26 00:00:00.000000000	200000.0	4000000000	102-Stuti -Upadhaya
1.	106	Vedshree	Jadhav	10	2004/03/17 00:00:00.000000000	50000.0	2500000000	106-Vedshree -Jadhav
1.	107	Aarti	Chauhan	10	2004/05/04 00:00:00.000000000	30000.0	9000000000	107-Aarti -Chauhan
1.	101	Susmita	Raul	10	2004/10/19 00:00:00.000000000	20000.0	4000000000	101-Susmita -Raul

Close



Spoon - Transformation 4 (changed)

Examine preview data

Rows of step: Concat fields (13 rows)

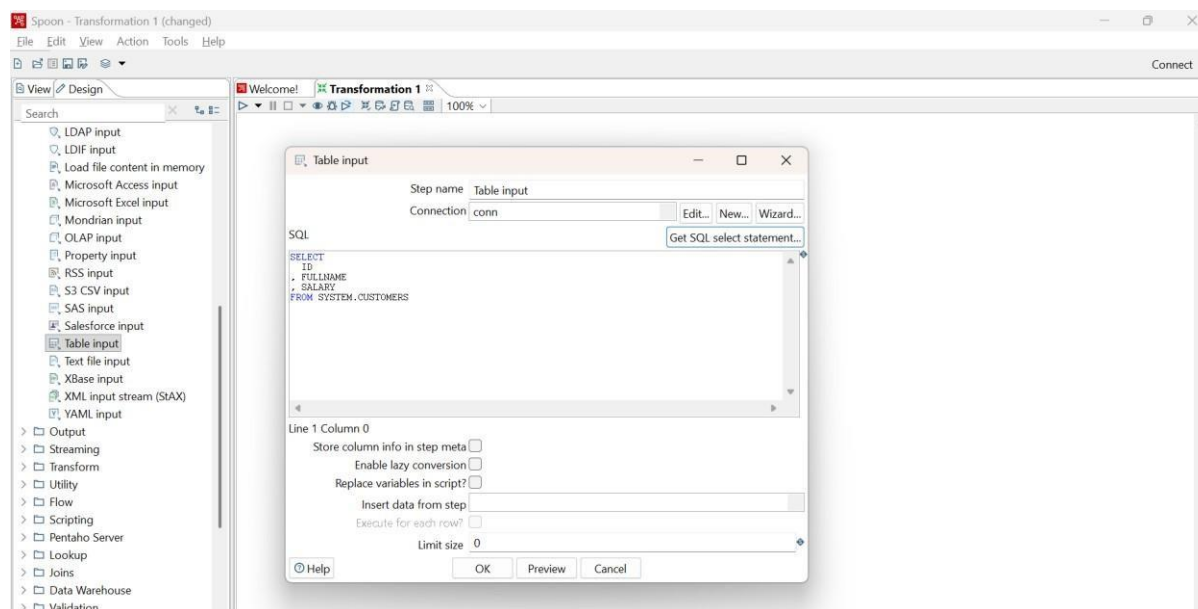
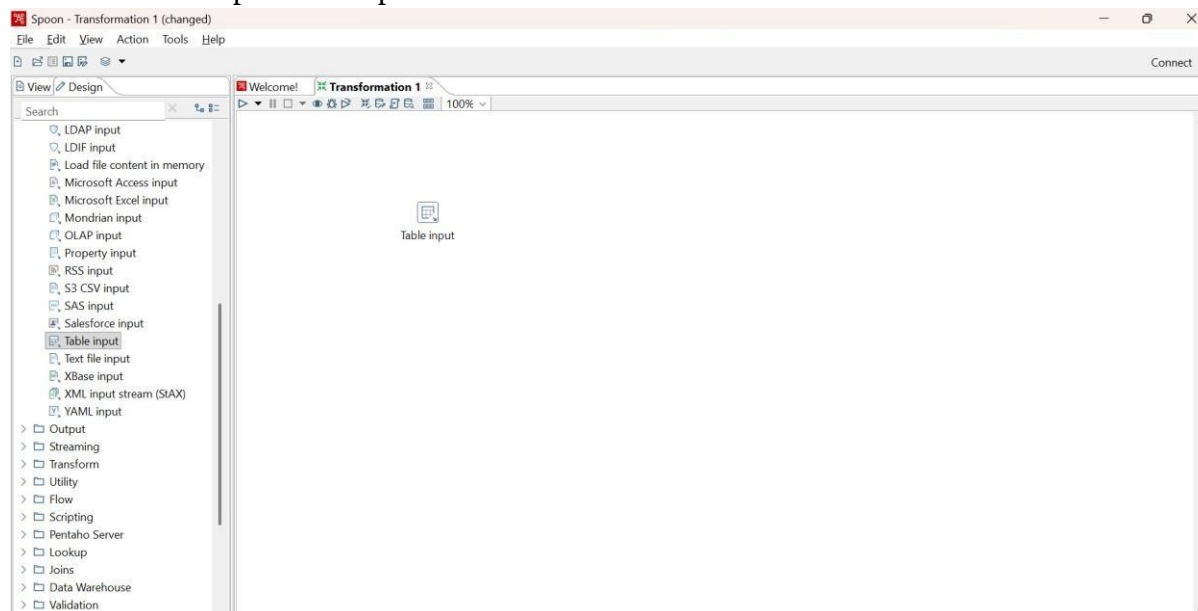
#	E_ID	FIRST_NAME	LAST_NAME	DEPT_ID	HIRE_DATE	SALARY	TOTALSAL	FULLNAME		
1	113	Chandan	Patil	30	2001/03/10 00:00:00.000000000	29000.0	841000000	113-Chandan	-Patil	-30-2001/03/10 00:00:00.000000000-00029000.00-841000000
2	110	Regina	Patil	30	2001/02/16 00:00:00.000000000	25000.0	625000000	110-Regina	-Patil	-30-2001/02/16 00:00:00.000000000-00025000.00-625000000
3	108	Mohammed	Singh	50	2001/01/19 00:00:00.000000000	30000.0	900000000	108-Mohammed	-Singh	-50-2001/01/19 00:00:00.000000000-00030000.00-900000000
4	118	Sreenivas	Patil	29	2001/01/29 00:00:00.000000000	20000.0	400000000	118-Sreenivas	-Patil	-29-2001/01/29 00:00:00.000000000-00020000.00-400000000
5	116	Naresh	Patil	20	2001/01/29 00:00:00.000000000	25000.0	625000000	116-Naresh	-Patil	-20-2001/01/29 00:00:00.000000000-00025000.00-625000000
6	126	Nataraj	Patil	40	2001/06/14 00:00:00.000000000	16000.0	256000000	126-Nataraj	-Patil	-40-2001/06/14 00:00:00.000000000-00016000.00-256000000
7	115	Prasanth	Patil	20	2001/07/16 00:00:00.000000000	6000.0	36000000	115-Prasanth	-Patil	-20-2001/07/16 00:00:00.000000000-00006000.00-36000000
8	104	Harshada	Deshmukh	10	2004/11/30 00:00:00.000000000	40000.0	1600000000	104-Harshada	-Deshmukh	-10-2004/11/30 00:00:00.000000000-00040000.00-1600000000
9	103	Sahil	Shah	10	2005/06/19 00:00:00.000000000	70000.0	4900000000	103-Sahil	-Shah	-10-2005/06/19 00:00:00.000000000-00070000.00-4900000000
1.	102	Stuti	Upadhaya	10	2004/07/26 00:00:00.000000000	200000.0	4000000000	102-Stuti	-Upadhaya	-10-2004/07/26 00:00:00.000000000-00200000.00-4000000000
1.	106	Vedshree	Jadhav	10	2004/03/17 00:00:00.000000000	50000.0	2500000000	106-Vedshree	-Jadhav	-10-2004/03/17 00:00:00.000000000-00050000.00-2500000000
1.	107	Aarti	Chauhan	10	2004/05/04 00:00:00.000000000	30000.0	900000000	107-Aarti	-Chauhan	-10-2004/05/04 00:00:00.000000000-00030000.00-900000000
1.	101	Susmita	Raul	10	2004/10/19 00:00:00.000000000	20000.0	400000000	101-Susmita	-Raul	-10-2004/10/19 00:00:00.000000000-00020000.00-400000000

Close

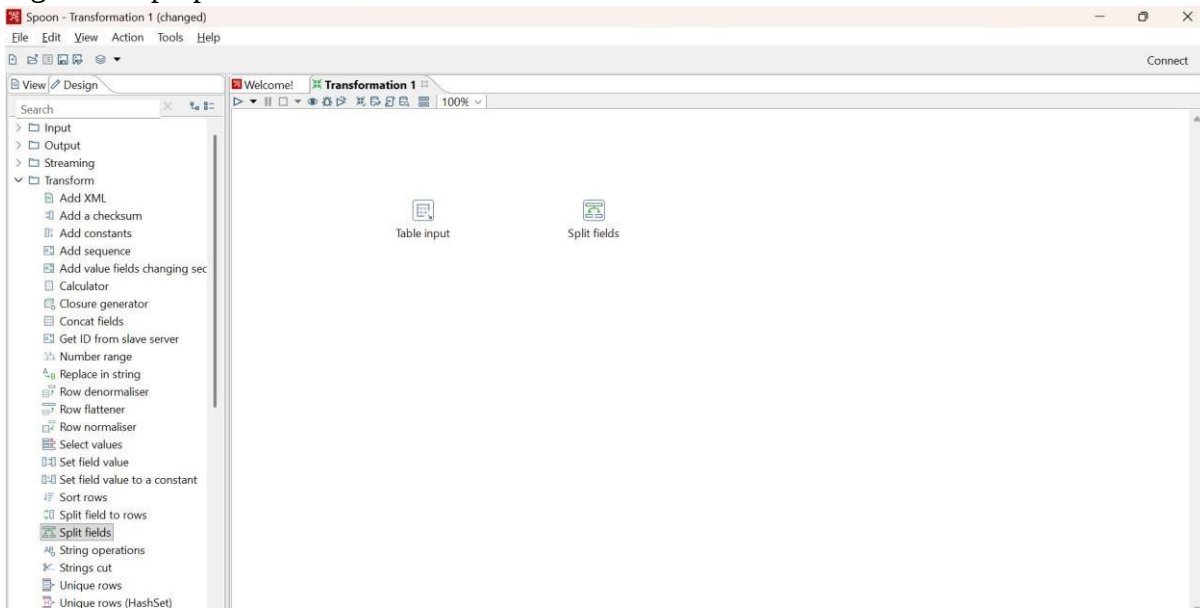
Split field to rows
Split fields
String operations
Strings cut
Unique rows

5. Splitting Operation

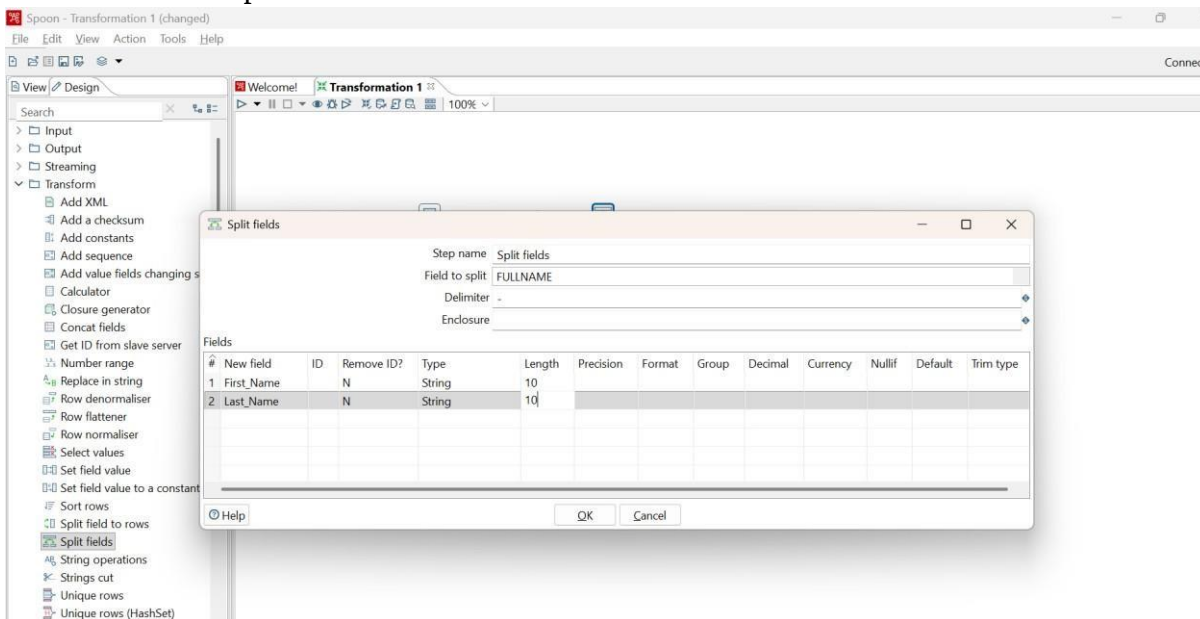
Perform first 6 steps same as practical 1.



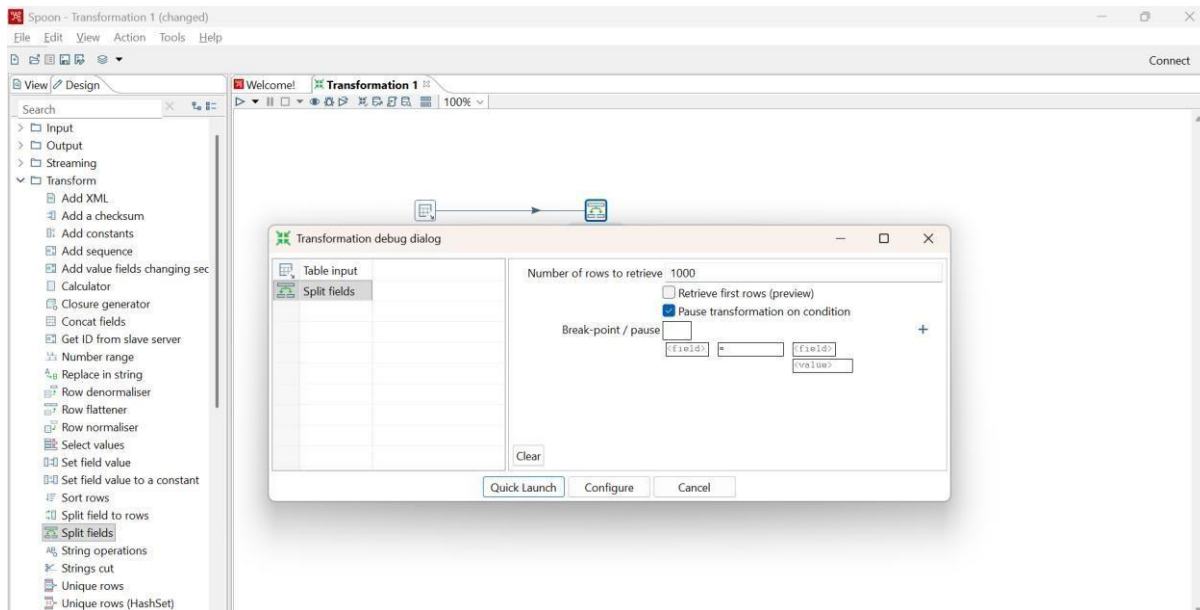
Drag and drop Split Field on the Panel



: Double Click On Split Field.



Click on Debug transformation and Click on Quick Launch.



Transformation 1

Table input → Split fields

Examine preview data

Rows of step: Split fields (5 rows)

#	EMPNO	First_Name	LastName	SALARY
1	205	David	Wilson	48000.0
2	204	Emily	Davis	70000.0
3	203	Michael	Johnson	55000.0
4	202	Jane	Smith	60000.0
5	201	John	Doe	50000.0

Spoon - Transformation 2 (changed)

File Edit View Action Tools Help

Connect

View Design

Search

- Input
- Output
 - Automatic documentation ou
 - Delete
 - Insert / update
 - JSON output
 - LDAP output
 - Microsoft Access output
 - Microsoft Excel output
 - Microsoft Excel writer
 - Pentaho reporting output
 - Properties output
 - RSS output
 - S3 file output
 - SQL file output
 - Salesforce delete
 - Salesforce insert
 - Salesforce update
 - Salesforce upsert
 - Serialize to file
 - Synchronize after merge
 - Table output
 - Text file output
 - Update
 - XML output
- Streaming
- Transform
- Utility

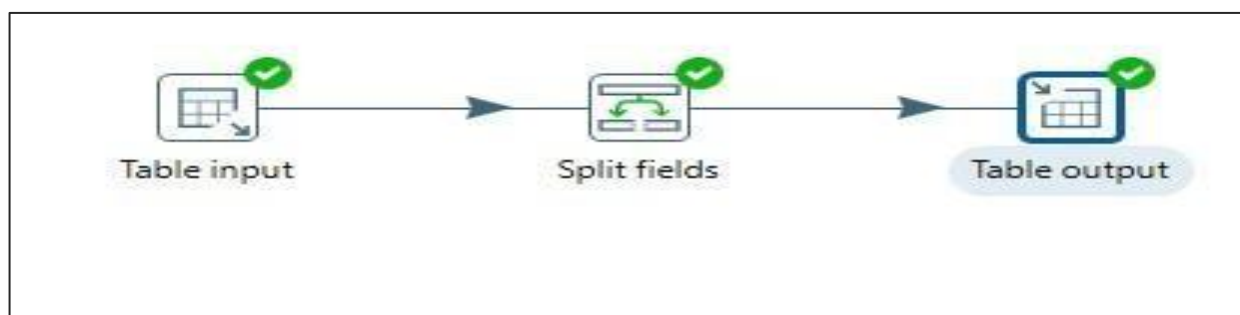
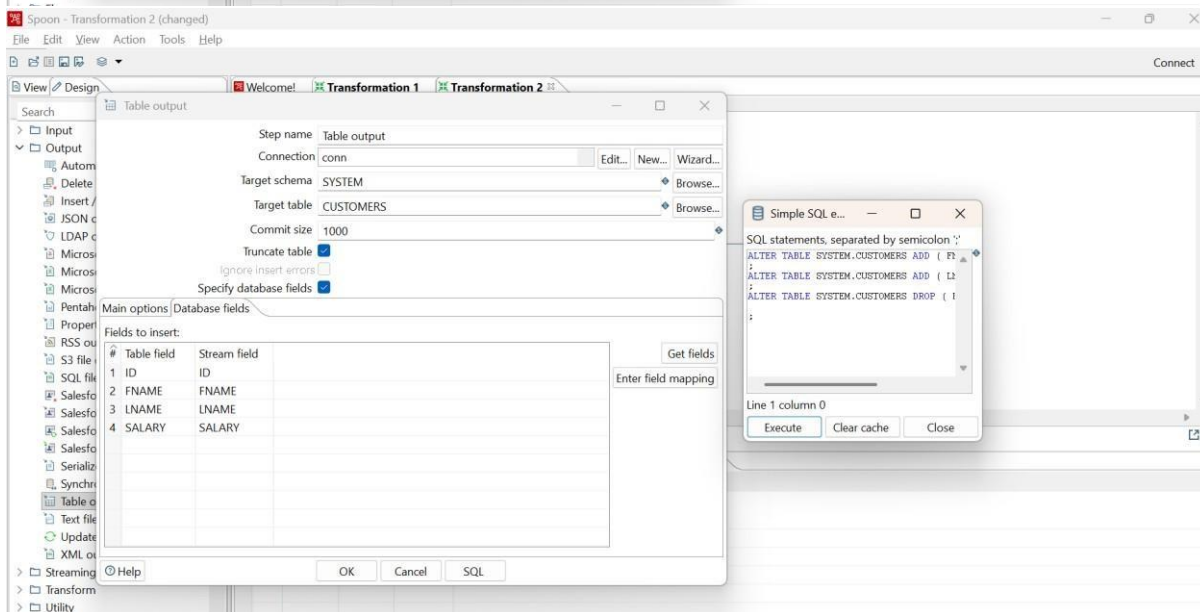
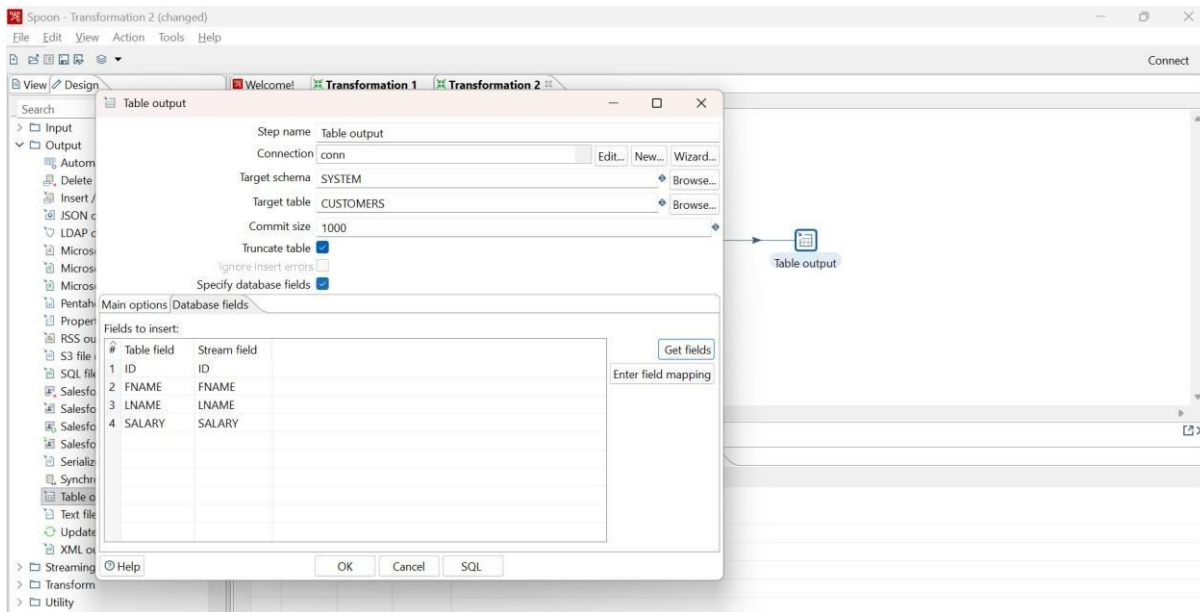
Transformation 2

Table input → Split fields → Table output

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

2025/11/11 23:12:16 - Transformation 2 - Dispatching started for transformation [Transformation 2]
2025/11/11 23:12:16 - Table input.0 - Finished reading query, closing connection
2025/11/11 23:12:16 - Spoon - The transformation has finished!!



SQL> select * from EMPLOYEE_DETAILS;

EMPNO	NAME	SALARY
201	John Doe	50000
202	Jane Smith	60000
203	Michael Johnson	55000
204	Emily Davis	70000
205	David Wilson	48000

SQL> select * from SplitTable;

EMPNO	FIRST_NAME	LASTNAME	SALARY
201	John	Doe	50000
202	Jane	Smith	60000
203	Michael	Johnson	55000
204	Emily	Davis	70000
205	David	Wilson	48000

6. Number Range:

```
SQL> -- Create the STUDENT table
SQL> CREATE TABLE STUDENT (
2     ROLL_NO NUMBER(3) PRIMARY KEY,
3     PERCENTAGE NUMBER(5,2)
4 );

Table created.

SQL>
SQL> -- Insert sample data
SQL> INSERT INTO STUDENT VALUES (101, 55);

1 row created.

SQL> INSERT INTO STUDENT VALUES (102, 59);

1 row created.

SQL> INSERT INTO STUDENT VALUES (103, 68);

1 row created.

SQL> INSERT INTO STUDENT VALUES (104, 75);

1 row created.

SQL> INSERT INTO STUDENT VALUES (105, 83);

1 row created.

SQL> INSERT INTO STUDENT VALUES (106, 35);

1 row created.

SQL> INSERT INTO STUDENT VALUES (107, 88);

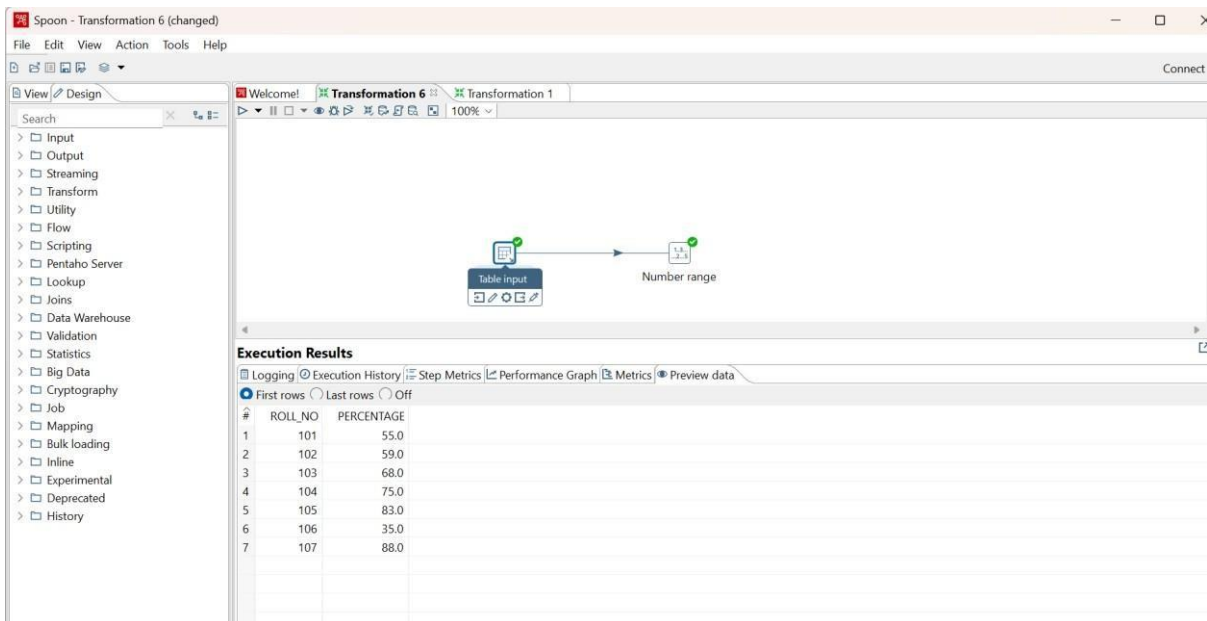
1 row created.

SQL> SELECT * FROM STUDENT;
```

ROLL_NO	PERCENTAGE
101	55
102	59
103	68
104	75
105	83
106	35
107	88

7 rows selected.

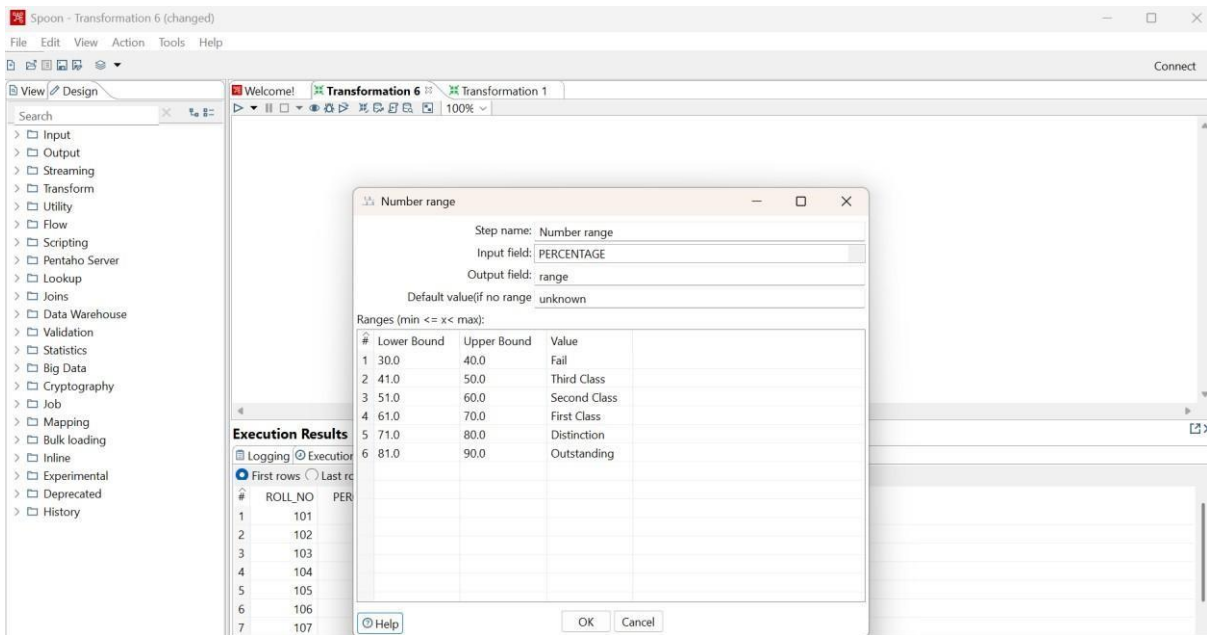
Perform first 6 steps same as practical 1.. Drag and drop Number Range transformation and double click on it.



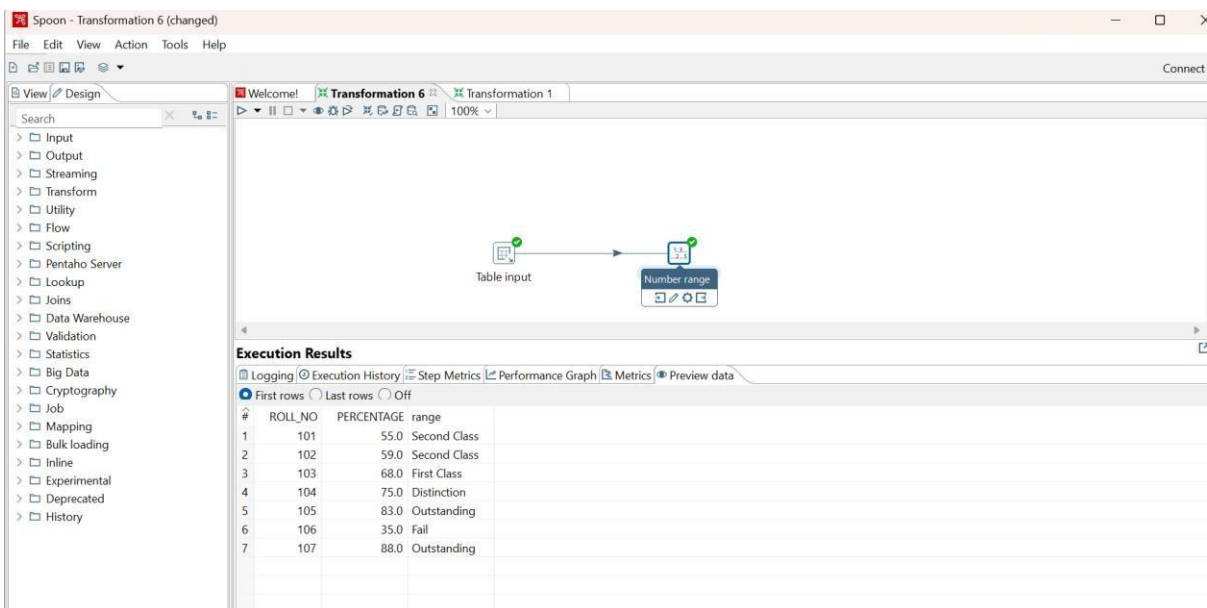
The screenshot shows the SAP Studio interface with a transformation job named 'Transformation 6 (changed)'. The job is composed of two transformations: 'Table Input' and 'Number range'. The 'Execution Results' pane at the bottom displays the output of the 'Number range' transformation, which is a table with 7 rows of data.

#	ROLL_NO	PERCENTAGE
1	101	55.0
2	102	59.0
3	103	68.0
4	104	75.0
5	105	83.0
6	106	35.0
7	107	88.0

Set values for Input field, Output field, Lower bound, Upper bound and value according to the values specified in percentage column



Click on Quick launch



7. String Operations

```

SQL> -- Create the NEWEMP table
SQL> CREATE TABLE NEWEMP (
2      EMPN      VARCHAR2(10) PRIMARY KEY,
3      ENAME     VARCHAR2(30),
4      DEPT      VARCHAR2(10),
5      JOB       VARCHAR2(20),
6      SALARY    NUMBER(10)
7 );

Table created.

SQL>
SQL> -- Insert sample employee data
SQL> INSERT INTO NEWEMP VALUES ('E001', 'John',      'D002', 'Manager', 50000);

1 row created.

SQL> INSERT INTO NEWEMP VALUES ('E002', 'Smit',      'D001', 'Hr',      40000);

1 row created.

SQL> INSERT INTO NEWEMP VALUES ('E003', 'steven_king', 'D002', NULL,      30000);

1 row created.

SQL> INSERT INTO NEWEMP VALUES ('E004', 'Lex Hean',   'D003', 'Manager', 14000);

1 row created.

SQL> INSERT INTO NEWEMP VALUES ('E005', 'Bruce',     'D003', 'Manager', 17000);

1 row created.

SQL> INSERT INTO NEWEMP VALUES ('E006', 'Kevin#Nash', 'D002', 'Sales',   18000);

1 row created.

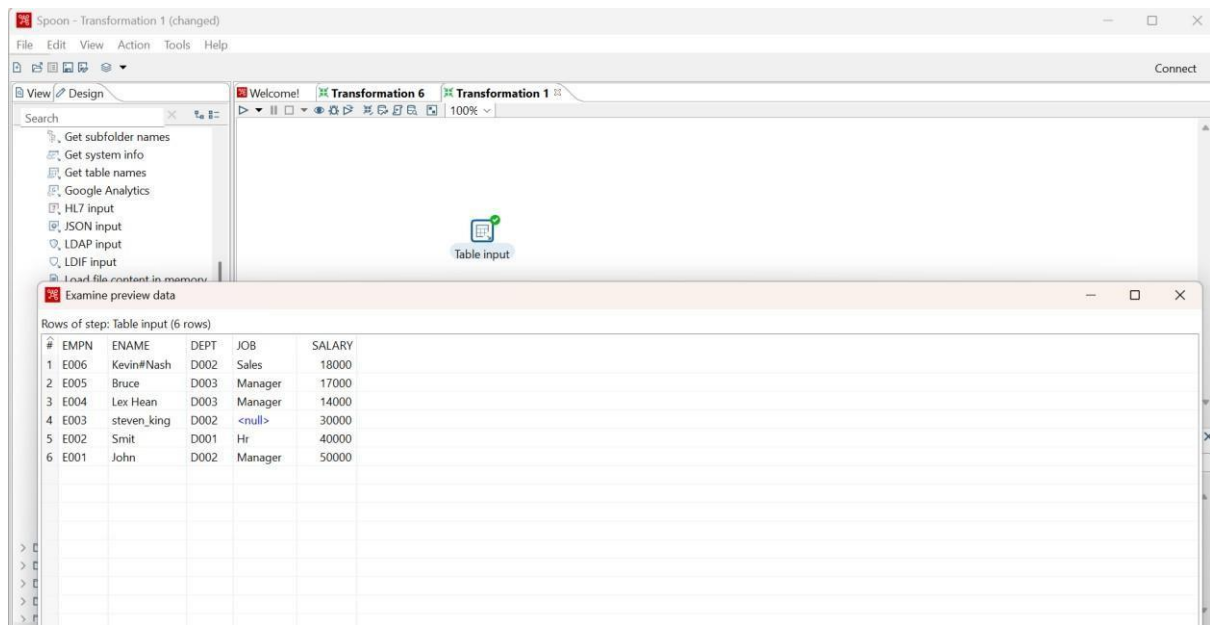
```

```
SQL> SELECT * FROM NEWEMP;
```

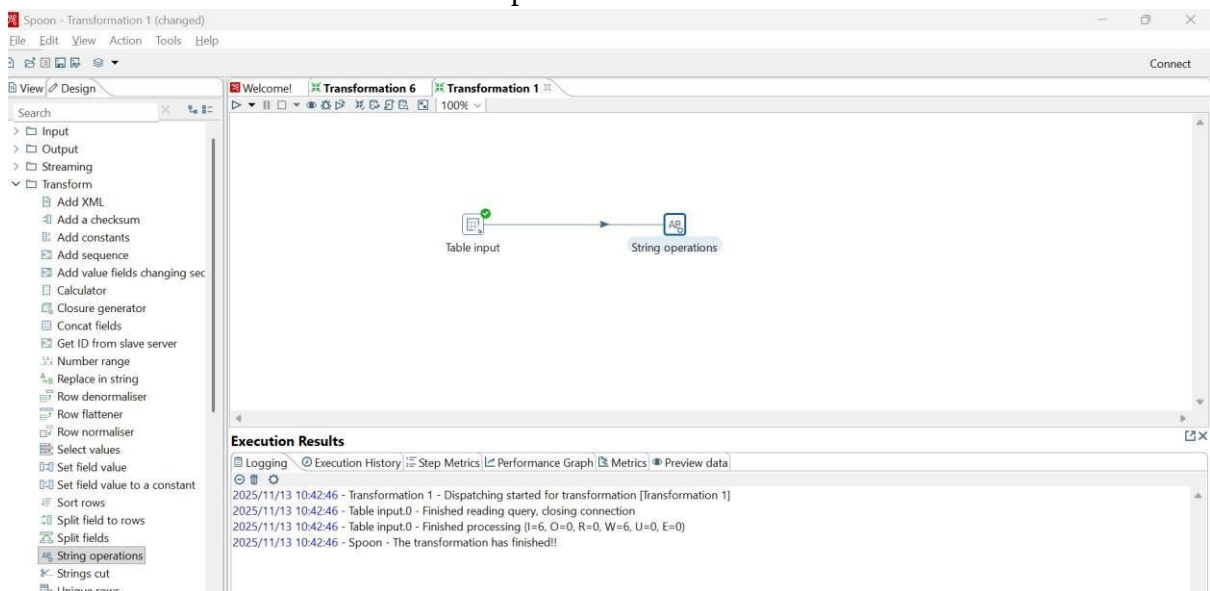
EMPN	ENAME	DEPT	JOB
E001	John	D002	Manager
E002	Smit	D001	Hr
E003	steven_king	D002	
E004	Lex Hean	D003	Manager
E005	Bruce	D003	Manager
E006	Kevin#Nash	D002	Sales

```
6 rows selected.
```

Perform first 6 steps same as practical 1.



Drag and drop String operation transformation, make connection between table input and string operation and double click on it..Set values for the parameters



Launch the transformation

The screenshot shows the SAP Data Services Spoon interface. The 'String operations' dialog box is open, displaying a table of fields to process. Below the dialog, the 'Execution Results' table is visible, showing the output of the transformation.

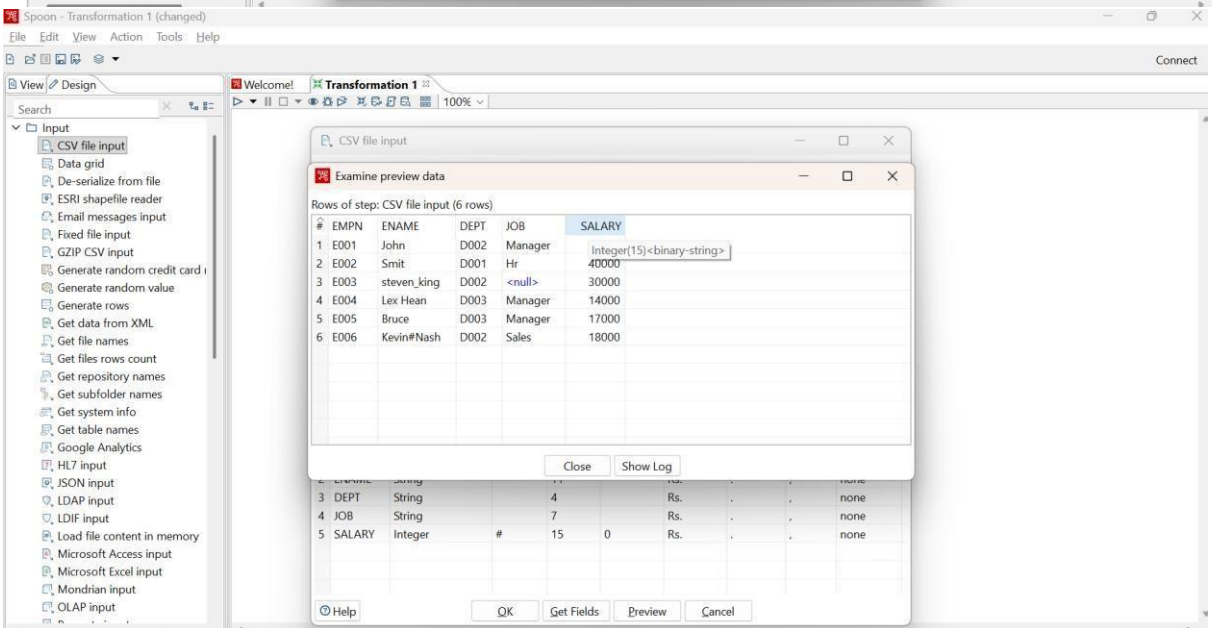
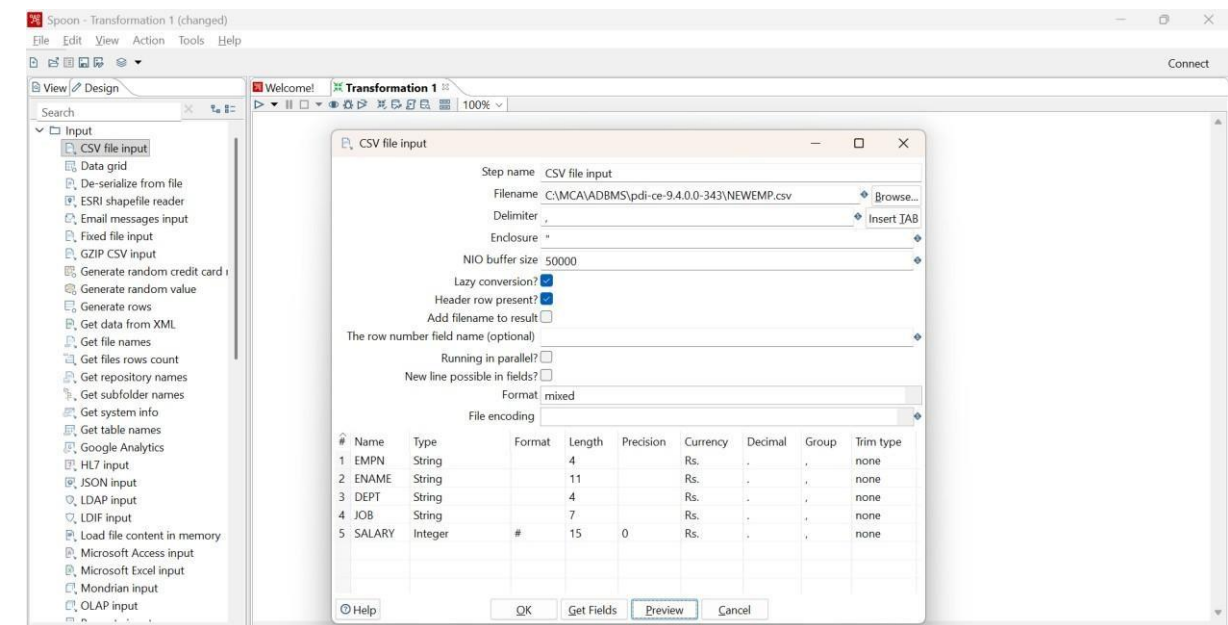
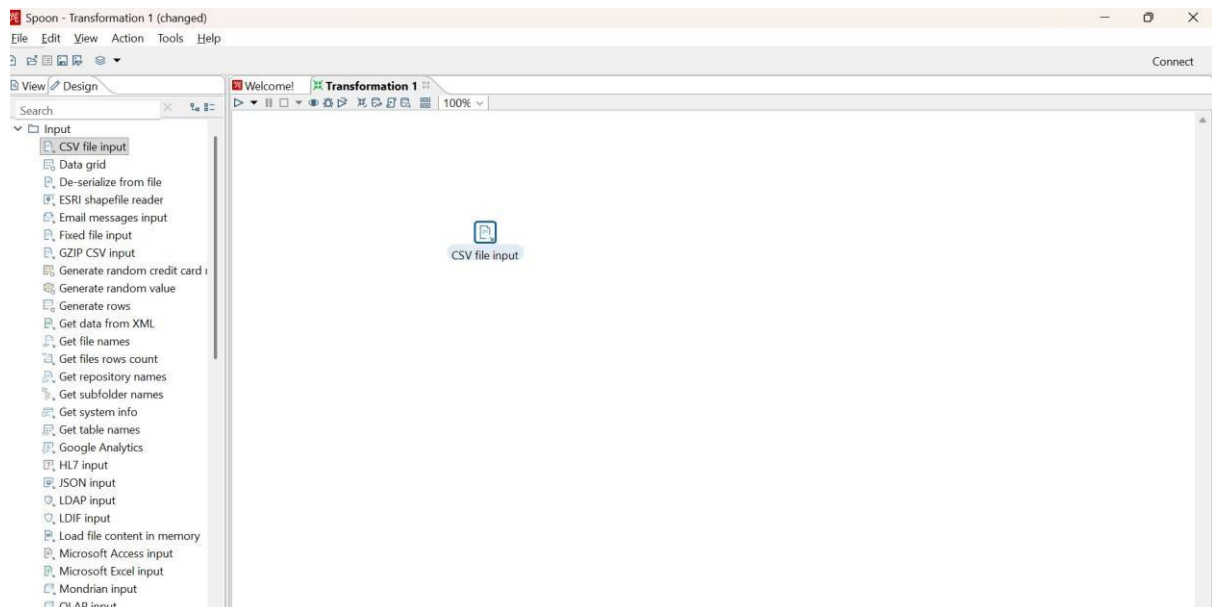
String operations dialog box:

#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char	Pad Length	InitCap	Escape	Digits	Remove Special character
1	EMPN		none	none	none			N	None	none	none
2	ENAME		none	none	none			N	None	none	horizontal tab
3	DEPT		none	none	none			N	None	none	none
4	JOB		none	none	none	*	10	N	None	none	none

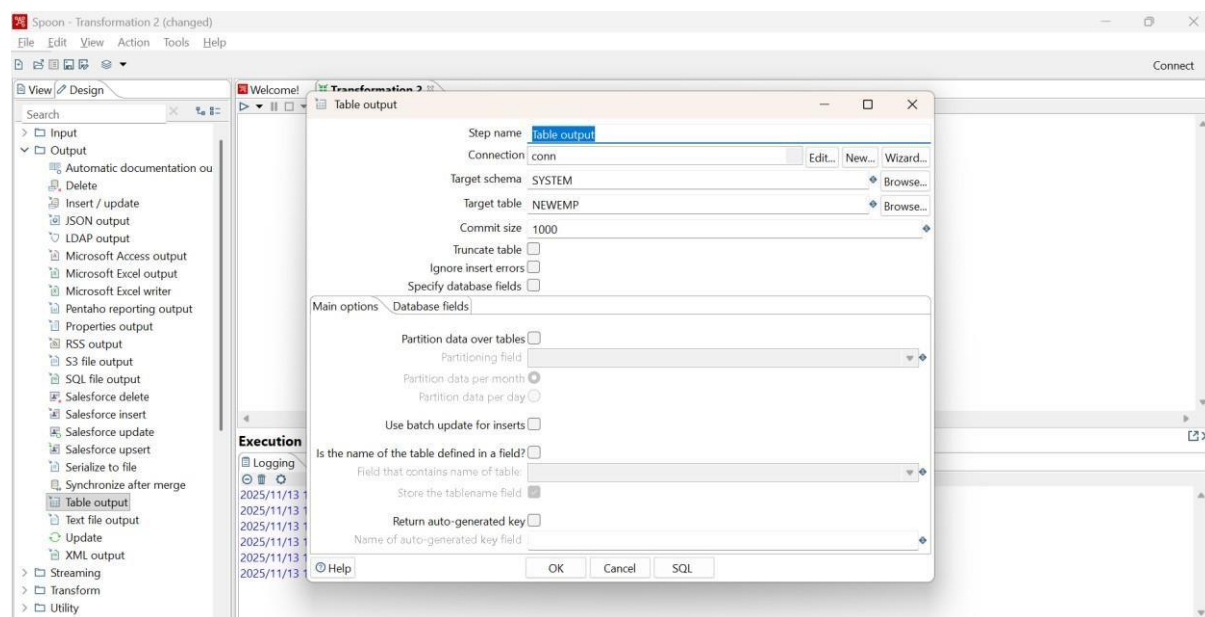
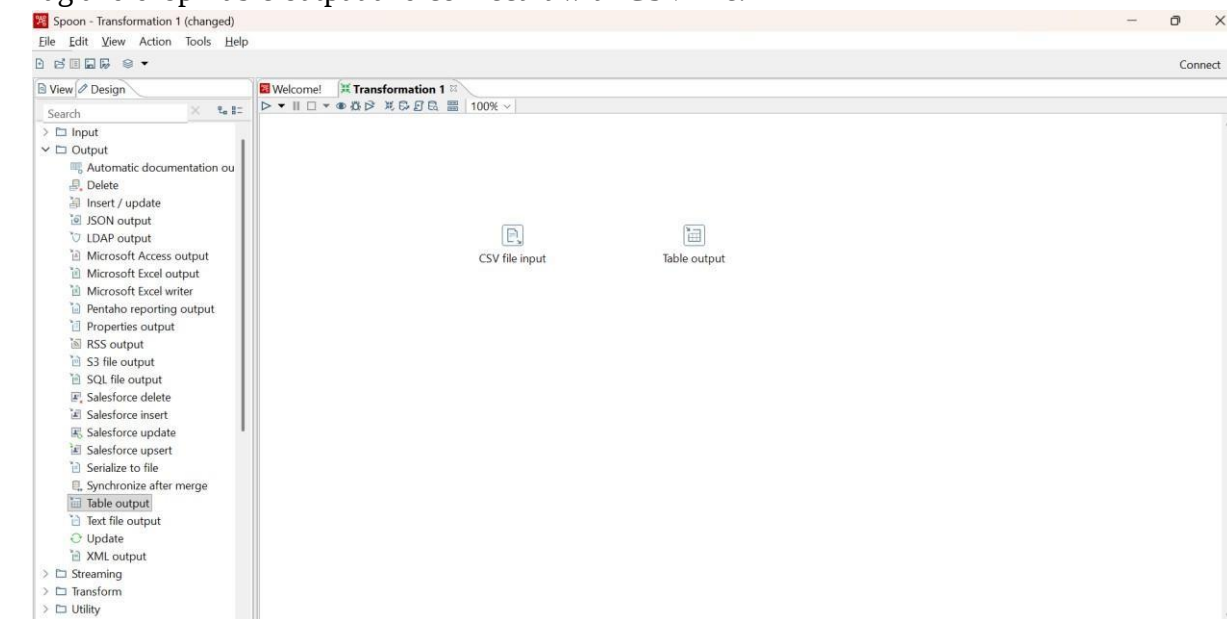
Execution Results:

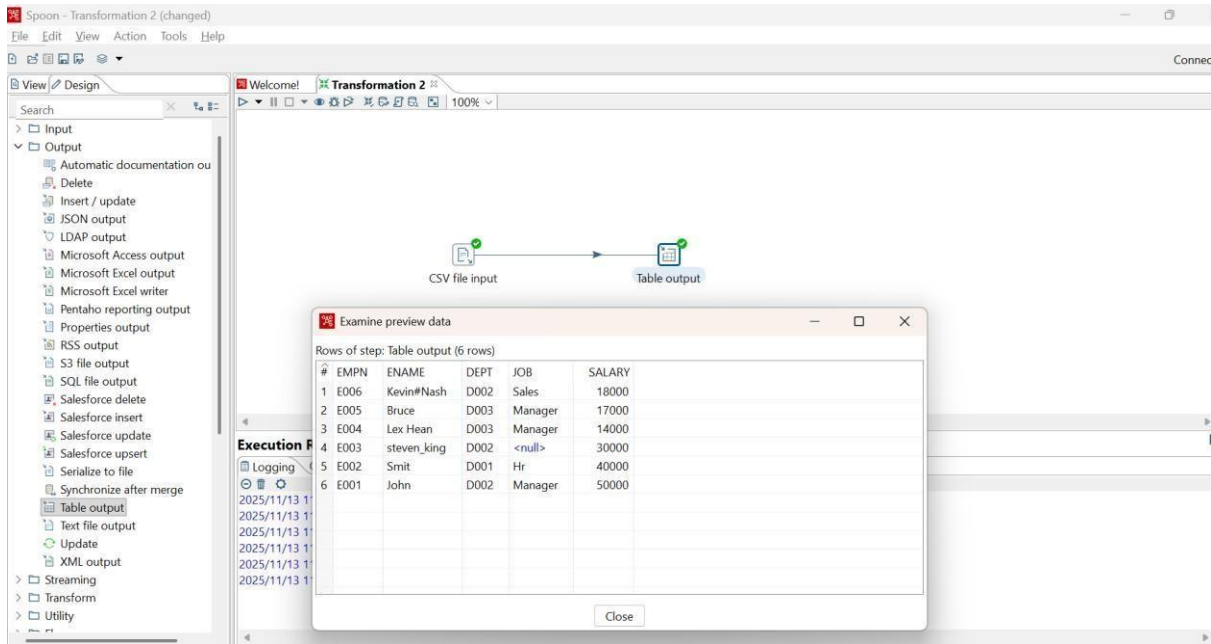
#	EMPN	ENAME	DEPT	JOB	SALARY
1	E001	John	D002	Manager	50000
2	E002	Smit	D001	Hr	40000
3	E003	steven_king	D002	<null>	30000
4	E004	Lex Hean	D003	Manager	14000
5	E005	Bruce	D003	Manager	17000
6	E006	Kevin#Nash	D002	Sales	18000

- Copy contents of.CSV file to the target table** Drag and drop CSV file input on a panel Double click on it and open respective CSV file, Lazy conversation and Options should be checked, click on Getfield and then ok.



Drag and drop Table output and connect it with CSV file.





```
SQL> SELECT * FROM NEWEMP;
```

EMPN	ENAME	DEPT	JOB

SALARY			

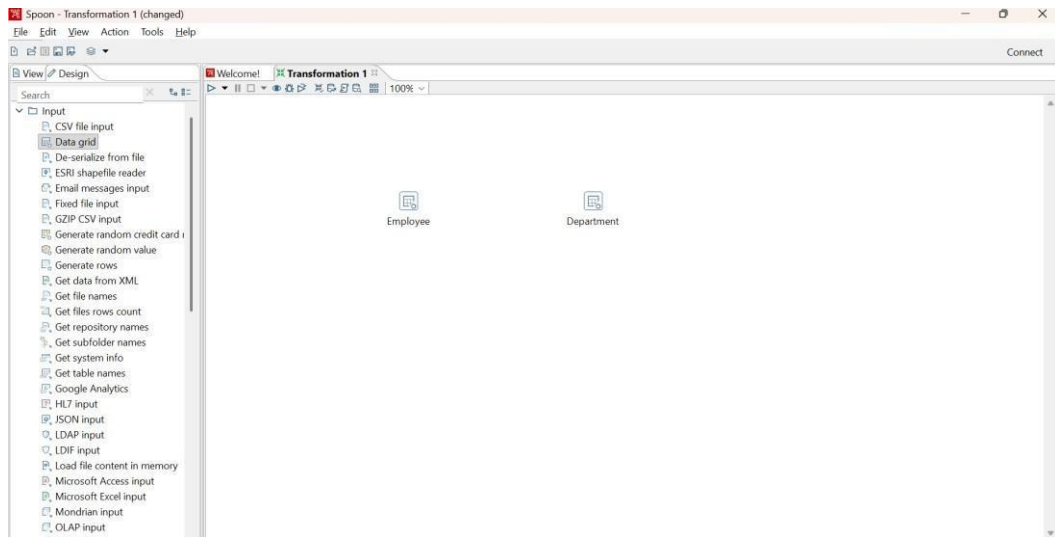
E001	John	D002	Manager
50000			
E002	Smit	D001	Hr
40000			
E003	steven_king	D002	
30000			

E004	Lex Hean	D003	Manager
14000			
E005	Bruce	D003	Manager
17000			
E006	Kevin#Nash	D002	Sales
18000			

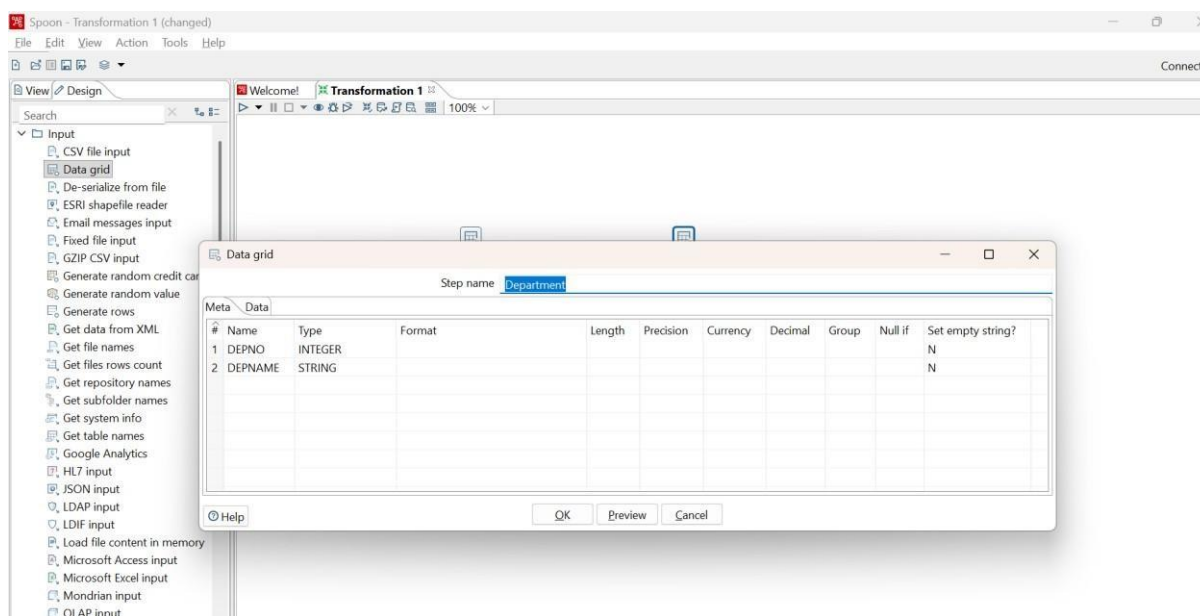
6 rows selected.

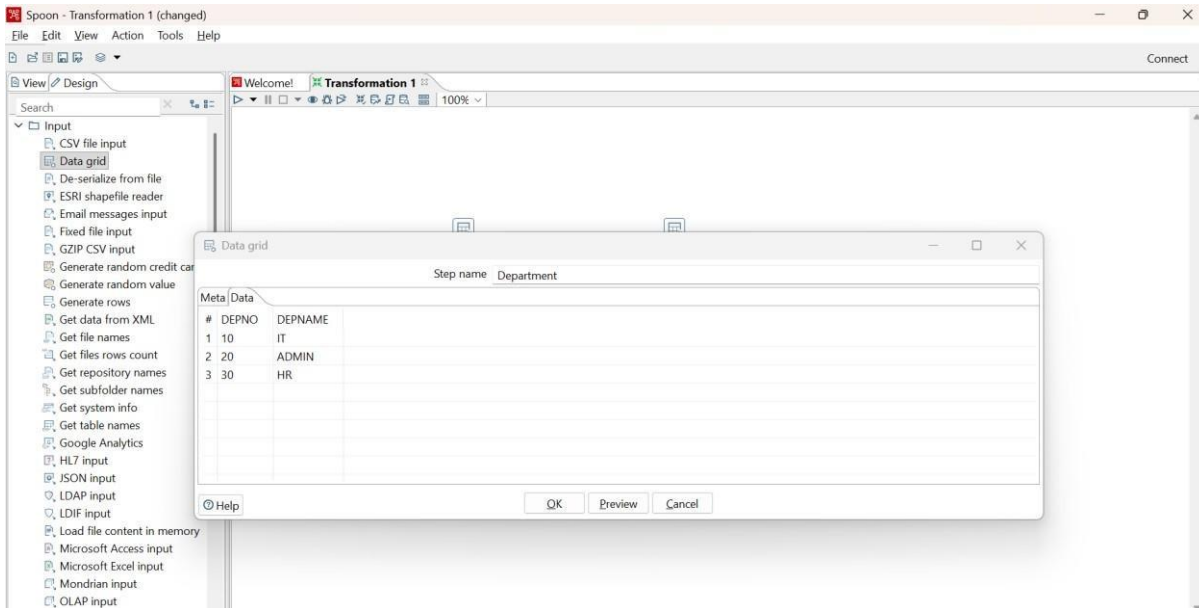
9. Implement the merge join transformation on tables

Drag two Data grid on panel rename them with Employee and Department respectively.

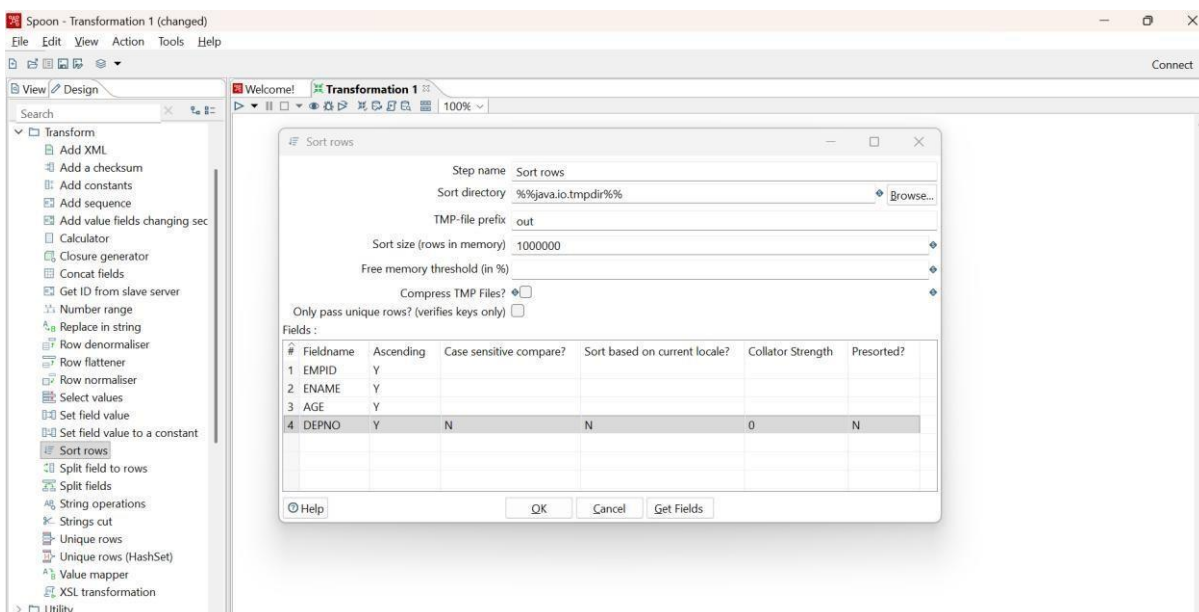
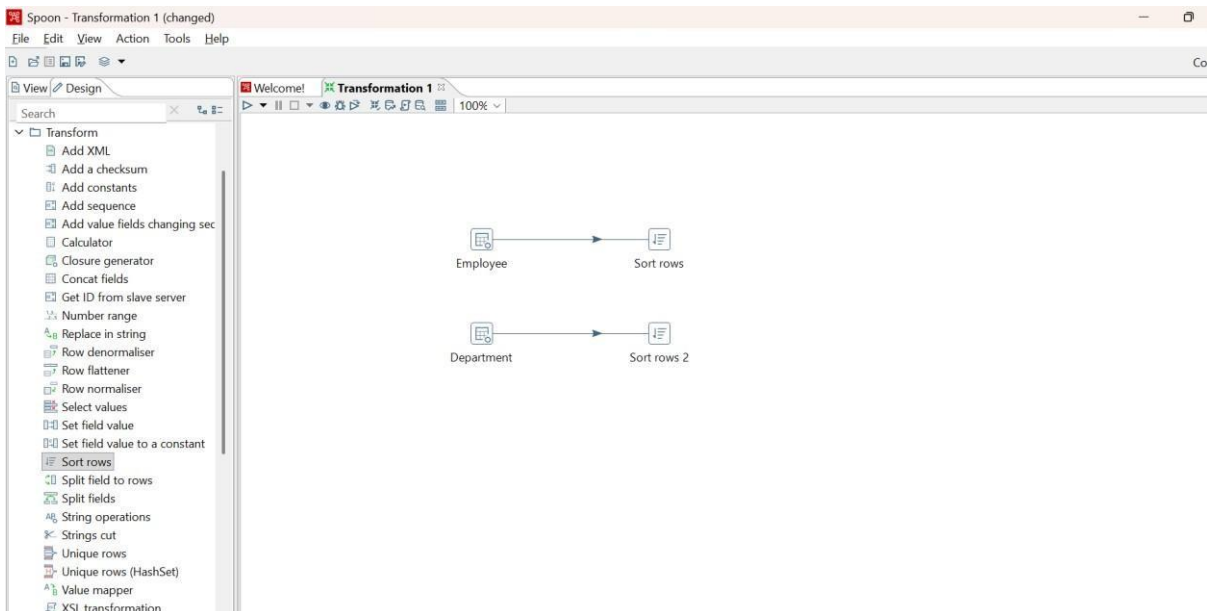


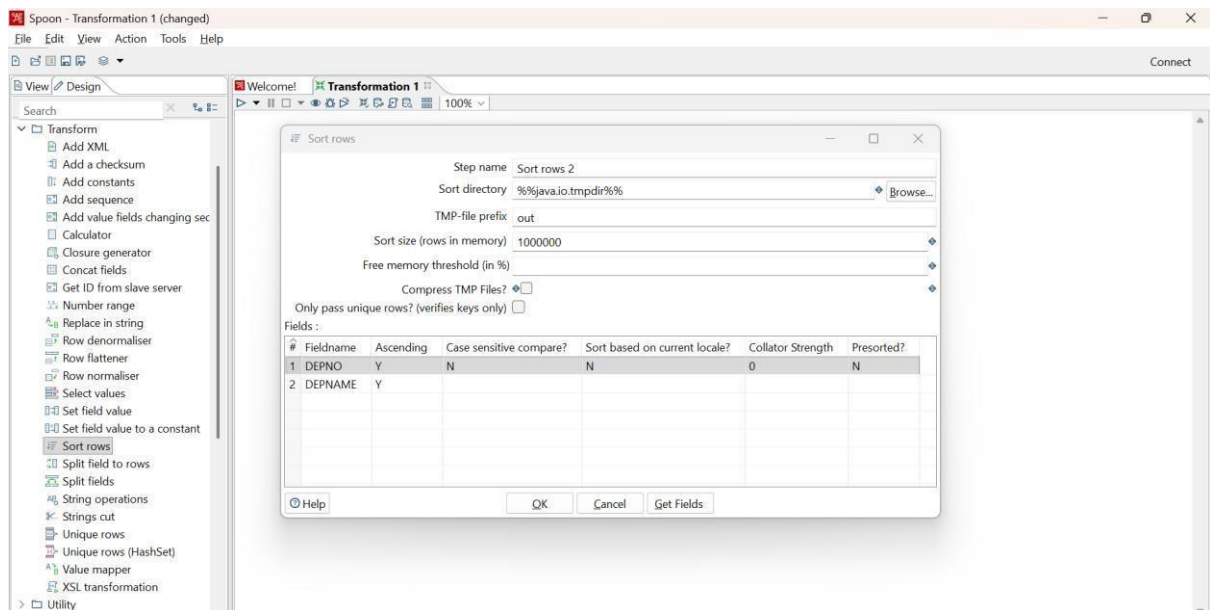
Insert
Records into respective grids.



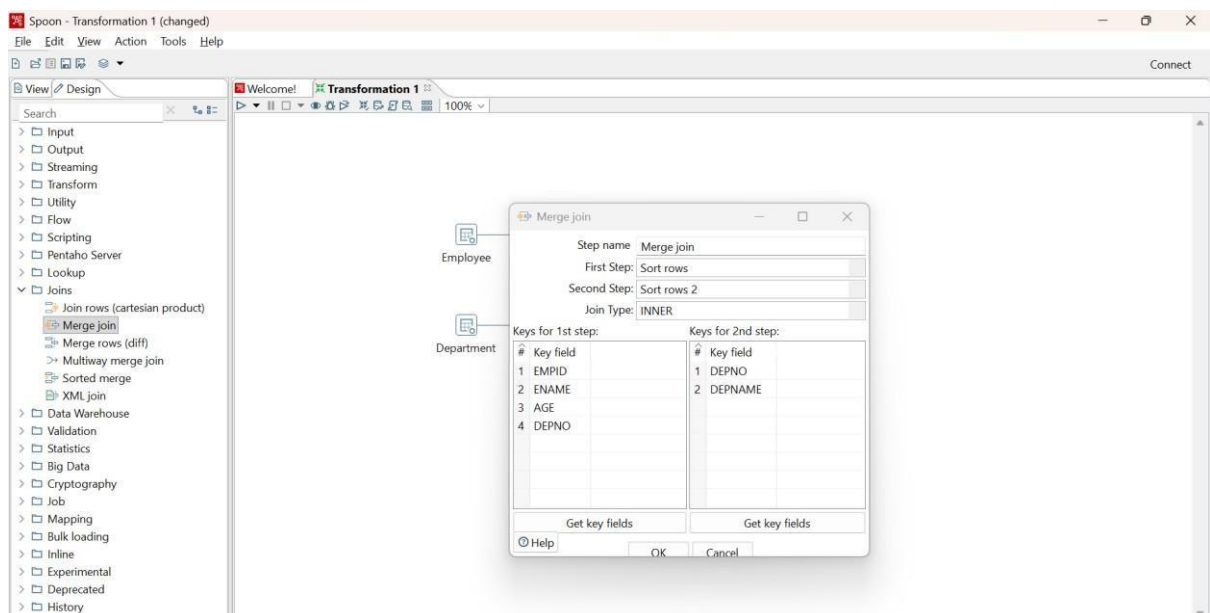
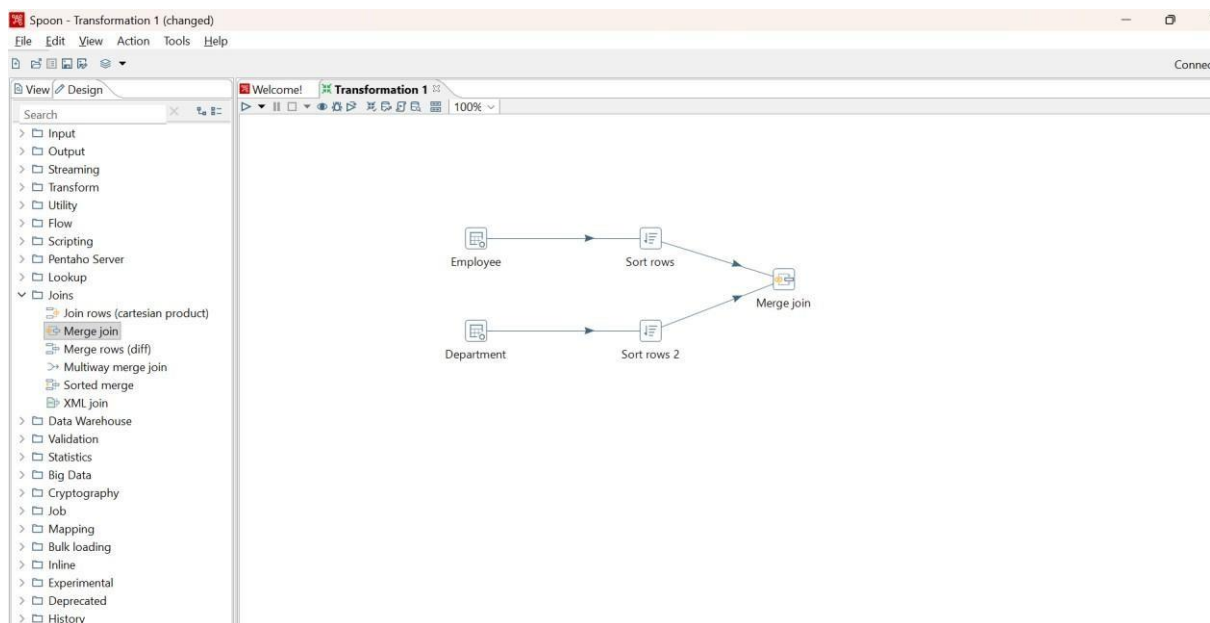


Insert sort rows transformation (on Empid) for both Employee and Department.

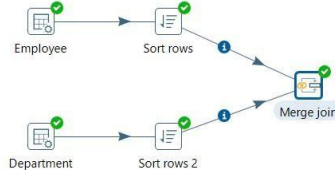




Add Merge join From Joins and connect it with sort rows.



Select Join Type Left Outer, then Right Outer and finally Full Outer join



Merge join

Step name Merge join

First Step: Sort rows

Second Step: Sort rows 2

Join Type: LEFT OUTER

Keys for 1st step: 1 Depno

Keys for 2nd step: 1 Depno

Get key fields

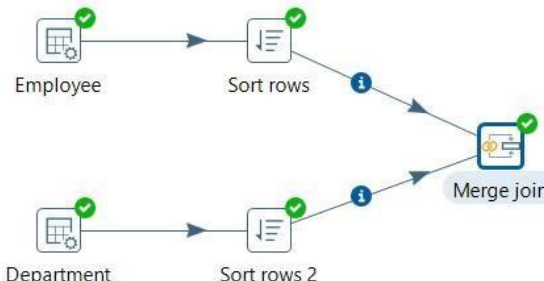
Get key fields

Help OK Cancel

Examine preview data

Rows of step: Merge join (5 rows)

#	Empid	Empname	Age	Depno	Depno_1	Depname
1	104	Ziya	17	40	<null>	<null>
2	103	Vanshika	22	30	30	Hr
3	102	Siddharth	21	20	20	Admin
4	101	Sarvesh	21	10	10	IT
5	<null>	<null>	<null>	<null>	<null>	<null>



Merge join

Step name Merge join

First Step: Sort rows

Second Step: Sort rows 2

Join Type: RIGHT OUTER

Keys for 1st step: 1 Depno

Keys for 2nd step: 1 Depno

Get key fields

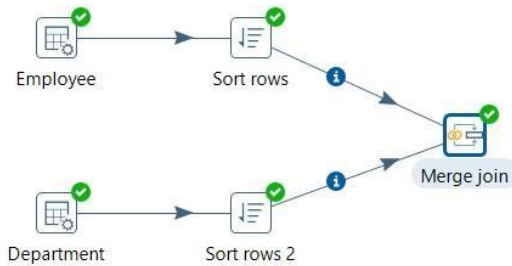
Get key fields

Help OK Cancel

Examine preview data

Rows of step: Merge join (3 rows)

#	Empid	Empname	Age	Depno	Depno_1	Depname
1	103	Vanshika	22	30	30	Hr
2	102	Siddharth	21	20	20	Admin
3	101	Sarvesh	21	10	10	IT



Merge join

Step name: Merge join

First Step: Sort rows

Second Step: Sort rows 2

Join Type: FULL OUTER

Keys for 1st step:

#	Key field
1	Depno

Keys for 2nd step:

#	Key field
1	Depno

Get key fields (for both steps)

Help OK Cancel

Examine preview data

Rows of step: Merge join (5 rows)

#	Empid	Empname	Age	Depno	Depno_1	Depname
1	104	Ziya	17	40	<null>	<null>
2	103	Vanshika	22	30	30	Hr
3	102	Siddharth	21	20	20	Admin
4	101	Sarvesh	21	10	10	IT
5	<null>	<null>	<null>	<null>	<null>	<null>

10. Implement different data validations on table.

Double click on Data grid and create product table in Meta tab and enter records into it using Data tab.

Data grid

Step name: Product

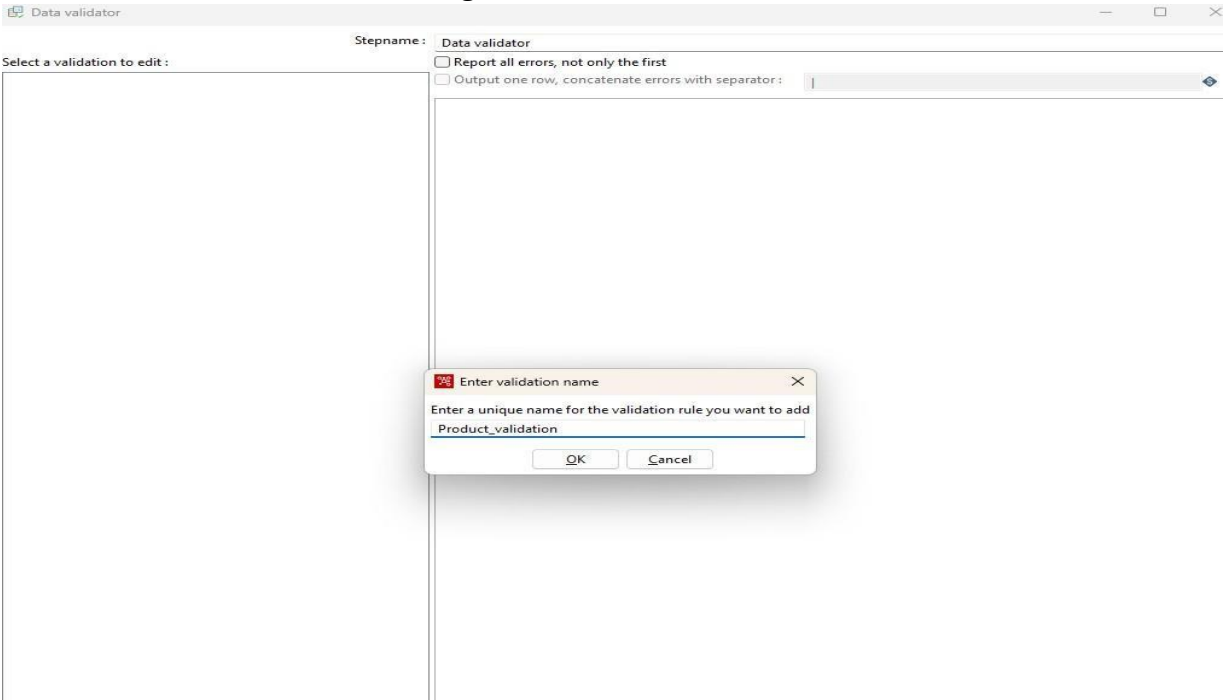
Meta Data

#	Name	Type	Format	Length	Precision	Cu
1	ProName	String				
2	ProPrice	integer				
3	ProId	integer				
4	Check	String				

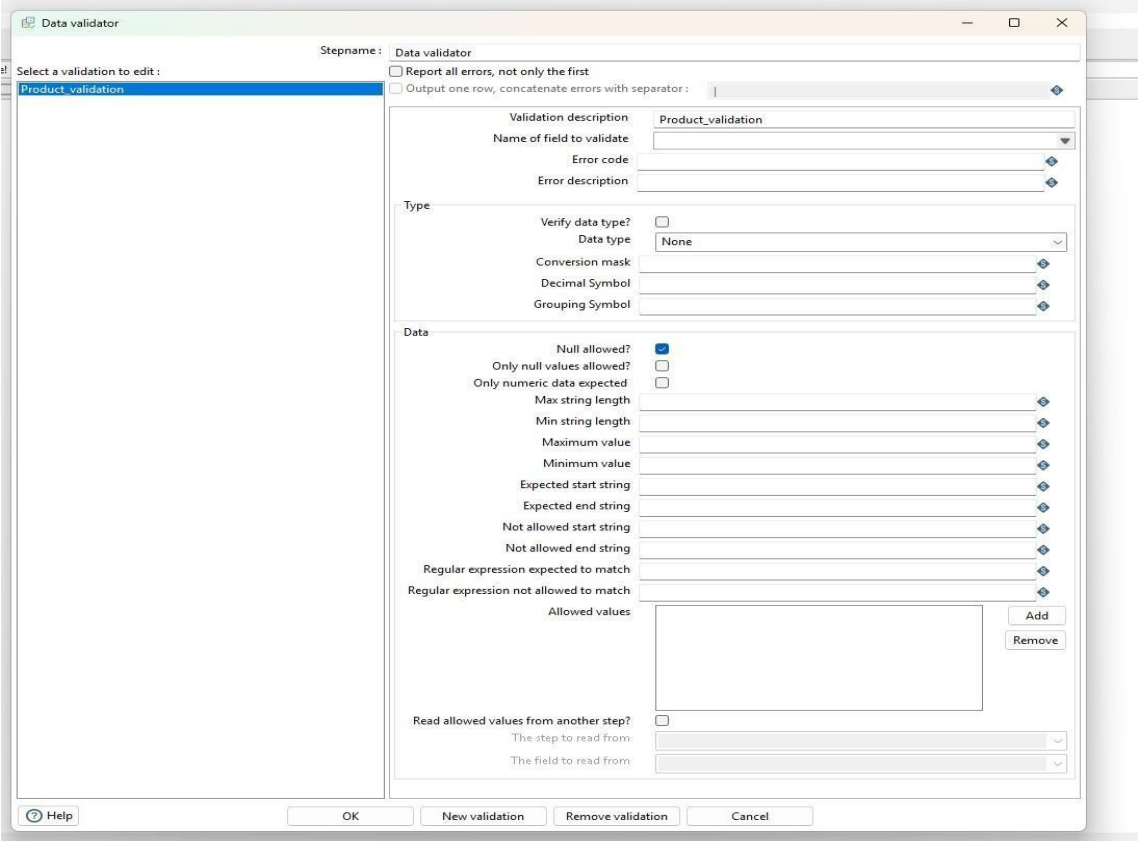
Go to Validation steps and select, Drag and Drop Data validator on panel and double click on it.



Click on new validation button and give new name to validation.



Once new validation is created it will appear on left panel, double click on it to set data validation



Set values for Name of field to validate=check status, and Data type=string

Data validator

Stepname: Data validator

Select a validation to edit: Product_validation

☐ Report all errors, not only the first

☐ Output one row, concatenate errors with separator: |

Validation description: Product_validation

Name of field to validate: Check status

Error code:

Error description:

Type

Verify data type? ☐

Data type: StringZ

Conversion mask:

Decimal Symbol:

Grouping Symbol:

Data

Null allowed? ☒

Only null values allowed? ☐

Only numeric data expected? ☐

Max string length:

Min string length:

Maximum value:

Minimum value:

Expected start string:

Expected end string:

Not allowed start string:

Not allowed end string:

Regular expression expected to match:

Regular expression not allowed to match:

Allowed values:

Add

Remove

Read allowed values from another step? ☐

The step to read from:

The field to read from:

Help OK New validation Remove validation Cancel

Click on add to set validation, set it to Shifted and press Enter and click on ok.

Data validator

Stepname: Data validator

Select a validation to edit: Product_validation

☐ Report all errors, not only the first

☐ Output one row, concatenate errors with separator: |

Validation description: Product_validation

Name of field to validate: Check status

Error code:

Error description:

Type

Verify data type? ☐

Data type: StringZ

Conversion mask:

Decimal Symbol:

Grouping Symbol:

Data

Null allowed? ☒

Only null values allowed? ☐

Only numeric data expected? ☐

Max string length:

Min string length:

Maximum value:

Minimum value:

Expected start string:

Expected end string:

Not allowed start string:

Not allowed end string:

Regular expression expected to match:

Regular expression not allowed to match:

Allowed values:

Add

Remove

Read allowed values from another step? ☐

The step to read from:

The field to read from:

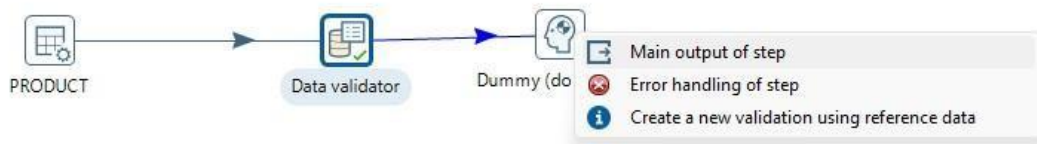
Help OK New validation Remove validation Cancel

Add allowed value

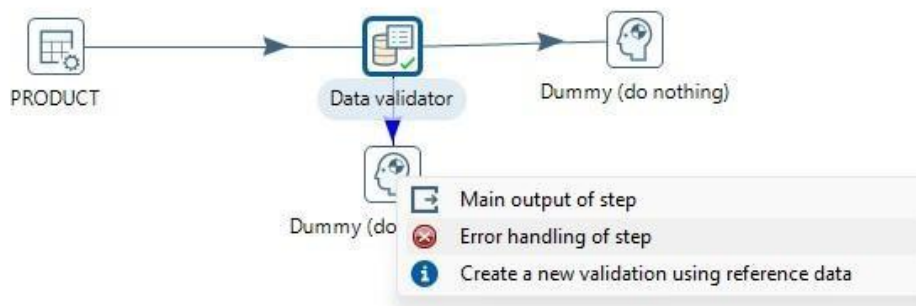
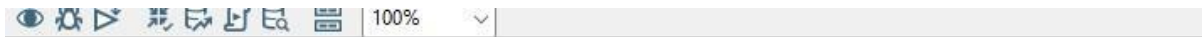
Enter the allowed value to add: shifted

OK Cancel

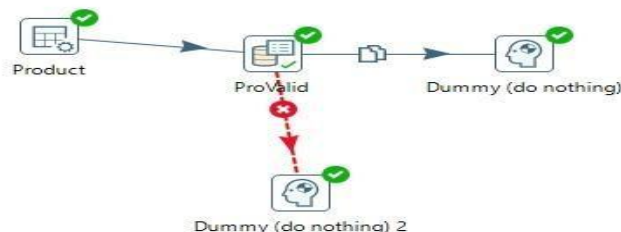
Select Dummy file from Flow to hold output and connect it with Data validation, while connecting select option "Main Output of step"



Select another dummy and connect it to ‘Validation for product’ and while selecting select ‘Error handling of step’ and in next window click on copy



Launch transformation by selecting one dummy file each.



Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

-12-14 14:10:07.848 - Transformation 6 - Dispatching started for transformation [Transformation 6]
 -12-14 14:10:07.858 - Product.0 - Finished processing (I=0, O=0, R=0, W=3, U=0, E=0)
 -12-14 14:10:07.863 - ProValid.0 - Finished processing (I=0, O=0, R=3, W=3, U=0, E=0)
 -12-14 14:10:07.866 - Dummy (do nothing).0 - Finished processing (I=0, O=0, R=3, W=3, U=0, E=0)
 -12-14 14:10:07.869 - Spoon - The transformation has finished!!

11. Replace Strings

Table input

Step name

Connection

SQL

```
SELECT
  ACCTNO
  CUSTNAME
  BRANCH
  ACCTBAL
FROM SYSTEM.ACCOUNTS
```

Replace in string

Step name

Fields string

#	In stream field	Out stream field	use RegEx	Search	Replace with	Set empty string?	Replace with file
1	CUSTNAME		N	sagar	Magar	N	

Table output

Step name

Connection

Target schema

Target table

Commit size

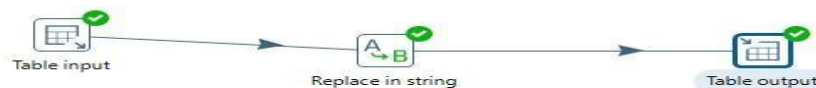
Truncate table ☒

Ignore insert errors ☐

Specify database fields ☒

Main options

Database fields



Execution Results

```

2024-12-14 14:15:59.335 - Transformation 7 - Dispatching started for transformation [Transformation 7]
2024-12-14 14:15:59.336 - Table output.0 - Connected to database [conn] (commit=1000)
2024-12-14 14:15:59.368 - Table input.0 - Finished reading query, closing connection
2024-12-14 14:15:59.369 - Table input.0 - Finished processing (I=7, O=0, R=0, W=7, U=0, E=0)
2024-12-14 14:15:59.372 - Replace in string.0 - Finished processing (I=0, O=0, R=7, W=7, U=0, E=0)
2024-12-14 14:15:59.382 - Table output.0 - Finished processing (I=0, O=7, R=7, W=7, U=0, E=0)
2024-12-14 14:15:59.383 - Spoon - The transformation has finished!!
  
```

Examine preview data

Rows of step: Replace in string (7 rows)

#	ACCTNO	CUSTNAME	BRANCH	ACCTBAL
7	1.0	Magar	mumbai	5000.0
6	2.0	arman	airport	9998.0
5	3.0	shashi	mumbai	7000.0
4	4.0	chutki	Fun-Fiesta	6000.0
3	7.0	sanju	ranchi	6500.0
2	8.0	monu	thane	11000.0
1	9.0	sonu	thane	11500.0

12.Splitting fields to Rows

Table input

Step name

Connection

SQL

```
SELECT
  ACCTNO
  CUSTNAME
  BRANCH
  ACCTBAL
  CONCAT
FROM SYSTEM.TARGET
```

Split field to rows

Step name

Field to split

Delimiter

Delimiter is a Regular ☐

New field name

Additional fields

Include rownum in output? ☐ Rownum fieldname

Reset Rownum at each input row? ☒

Table output

Step name

Connection

Target schema

Target table

Commit size

Truncate table ☒

Ignore insert errors ☐

Specify database fields ☒



Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data



2024-12-14 14:21:47.376 - Transformation 8 - Dispatching started for transformation [Transformation 8]
 2024-12-14 14:21:47.378 - Table output.0 - Connected to database [conn] (commit=1000)
 2024-12-14 14:21:47.411 - Table input.0 - Finished reading query, closing connection
 2024-12-14 14:21:47.412 - Table input.0 - Finished processing (I=42, O=0, R=0, W=42, U=0, E=0)
 2024-12-14 14:21:47.415 - Split field to rows.0 - Finished processing (I=0, O=0, R=42, W=49, U=0, E=0)
 2024-12-14 14:21:47.425 - Table output.0 - Finished processing (I=0, O=49, R=49, W=49, U=0, E=0)
 2024-12-14 14:21:47.427 - Spoon - The transformation has finished!!

Examine preview data

Rows of step: Split field to rows (49 rows)

#	ACCTNO	CUSTNAME	BRANCH	ACCTBAL	CONCAT	Bal
1	9.0	sonu	thane	11500.0	9.0-11500.0	9.0-11500.0
2	8.0	monu	thane	11000.0	8.0-11000.0	8.0-11000.0
3	7.0	sanju	ranchi	6500.0	7.0-6500.0	7.0-6500.0
4	4.0	chutki	Fun-Fiesta	6000.0	4.0-6000.0	4.0-6000.0
5	3.0	shashi	mumbai	7000.0	3.0-7000.0	3.0-7000.0
6	2.0	arman	airport	9998.0	2.0-9998.0	2.0-9998.0
7	1.0	sagar	mumbai	5000.0	1.0-5000.0	1.0-5000.0
8	9.0	sonu	thane	11500.0	9.0-11500.0	9.0-11500.0
9	8.0	monu	thane	11000.0	8.0-11000.0	8.0-11000.0
10	7.0	sanju	ranchi	6500.0	7.0-6500.0	7.0-6500.0
11	4.0	chutki	Fun-Fiesta	6000.0	4.0-6000.0	4.0-6000.0
12	3.0	shashi	mumbai	7000.0	3.0-7000.0	3.0-7000.0
13	2.0	arman	airport	9998.0	2.0-9998.0	2.0-9998.0
14	1.0	sagar	mumbai	5000.0	1.0-5000.0	1.0-5000.0
15	9.0	sonu	thane	11500.0	9.0;11500.0	11500.0
16	9.0	sonu	thane	11500.0	9.0;11500.0	9.0
17	8.0	monu	thane	11000.0	8.0;11000.0	11000.0
18	8.0	monu	thane	11000.0	8.0;11000.0	8.0
19	7.0	sanju	ranchi	6500.0	7.0;6500.0	6500.0
20	7.0	sanju	ranchi	6500.0	7.0;6500.0	7.0

PRACTICAL NO 5

Introduction to R, Install packages, Loading packages Data types, checking variable type, printing variable and objects (Vector, Matrix, List, Factor, Data frame, Table) c-binding and rebinding

Reading and Writing data

Setwd(), getwd(), data(), rm()

Attaching and Detaching data

Reading data from the console Loading data from different data sources (CSV, Excel)

Assigning Variable

```
## Assigning variable
```

```
var1<-2
```

```
var2<-3
```

```
var3<-7
```

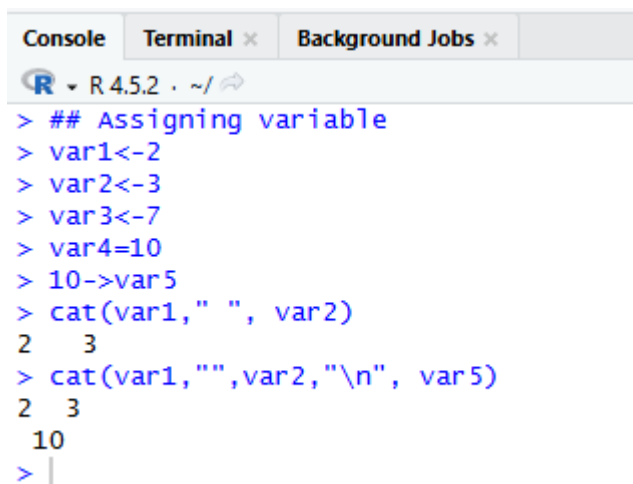
```
var4=10
```

```
10->var5
```

```
cat(var1," ", var2)
```

```
cat(var1,"",var2,"\n", var5)
```

Output –



```
Console Terminal x Background Jobs x
R R 4.5.2 ~ /
> ## Assigning variable
> var1<-2
> var2<-3
> var3<-7
> var4=10
> 10->var5
> cat(var1," ", var2)
2 3
> cat(var1,"",var2,"\n", var5)
2 3
10
> |
```

Basics of R

```
5+6
```

```
sum<- 6+7
```

```
sum
```



```
print(sum)
```

```
##my program
```

```
demo<- 2+3 ;demo1<-5+6
```

```
print(demo)
```

```
print(demo1)
```

```
sessionInfo()
```

```
x<-1:10
```

```
plot(x)
```

```
a=5
```

```
A=10
```

```
a
```

```
A
```

```
R 4.5.2 - ~/
> 5+6
[1] 11
> sum<- 6+7
> sum
[1] 13
> print(sum)
[1] 13
> ##my program
> demo<- 2+3 ;demo1<-5+6
> print(demo)
[1] 5
> print(demo1)
[1] 11
> sessionInfo()
R version 4.5.2 (2025-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: windows 10 x64 (build 19045)

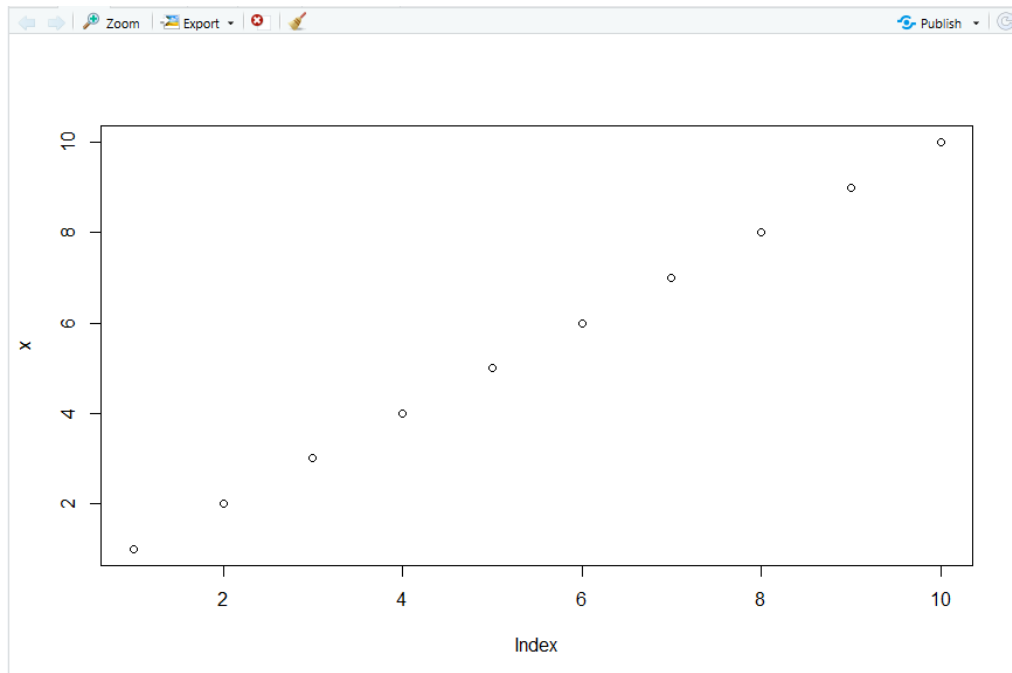
Matrix products: default
LAPACK version 3.12.1

locale:
[1] LC_COLLATE=English_India.utf8 LC_CTYPE=English_India.utf8 LC_MONETARY=English_India.utf8
[4] LC_NUMERIC=C LC_TIME=English_India.utf8

time zone: Asia/Calcutta
tzcode source: internal

attached base packages:
[1] stats graphics grDevices utils datasets methods base

loaded via a namespace (and not attached):
[1] compiler_4.5.2 Matrix_1.7-4 generics_0.1.4 tools_4.5.2 arules_1.7-11 grid_4.5.2 lattice_0.22-7
> x<-1:10
> plot(x)
> a=5
> A=10
> a
[1] 5
> A
[1] 10
> |
```

Data Types in R

Data types in R

logical, numeric, integer, complex, logical, character

num<- 12,12.35,-34.45,11.67

integer<-36L

complex<-5+2i

logical= TRUE,FALSE (not 0 and 1)

character= 'a',"Hello R", "FALSE",'26.3454'

num<- 12

class(num)

intl<-15

class(intl)

intl<-as.integer(intl)

class(intl)

int2<-45L

class(int2)

R 4.5.2 · ~/

```
> ## Data types in R
> ## logical, numeric, integer, complex, logical, character
> ## num<- 12,12.35,-34.45,11.67
> ## integer<-36L
> ## complex<-5+2i
> ## logical= TRUE,FALSE ( not 0 and 1)
> ## character= 'a',"Hello R", "FALSE",'26.3454'
> num<- 12
> class(num)
[1] "numeric"
> int1<-15
> class(int1)
[1] "numeric"
> int1<-as.integer(int1)
> class(int1)
[1] "integer"
> int2<-45L
> class(int2)
[1] "integer"
> |
```

c-binding and rbinding

R Matrix

Matrix is two dimensional data

##matrix(data,nrow,ncol,byrow,dim_name)

##mat<-matrix(c(2:13),nrow=4,byrow = TRUE)

##mat<-matrix(c(2,3,11,5,6,7),nrow= 3, ncol= 2, byrow= TRUE)

##mat

##mat<-matrix(c(2,3,11,5,6,7),nrow= 2, ncol= 3, byrow= TRUE)

##mat

x<-matrix(c(5:16), nrow=4, byrow= TRUE)

y<-matrix(c(7:18),nrow=4, byrow= FALSE)

x

y

Naming Matrix

row_name<-c("r1","r2","r3","r4")

col_names<-c("c1","c2","c3")

```
z<- matrix(c(7:18), nrow=4, byrow= TRUE , dimnames= list(row_name,col_names))
```

```
z
```

```
## Accessing elements
```

```
print(z[3,1])
```

```
print(z[,1])
```

```
print(z[2,])
```

```
## updating elements
```

```
z[4,3]<-0
```

```
z[z==11]<-0
```

```
z[z>15]<-0
```

```
#cbind() and rbind() are used to add col n row resp
```

```
rbind(z,c(2,3,4))
```

```
cbind(z,c(8,5,2,0))
```

```
z
```

```
## Transpose i.e row into col n col into row
```

```
t(z)
```

```
a1<-matrix(c(5:16), nrow=4, ncol=3)
```

```
a2<-matrix(c(1:12), nrow=4, ncol=3)
```

```
sum<-a1+a2
```

```
sum
```

R v R 4.5.2 . ~/

```
> # R Matrix
> ## Matrix is two dimensional data
> ##matrix(data,nrow,ncol,byrow,dim_name)
> ##mat<-matrix(c(2:13),nrow=4,byrow = TRUE)
> ##mat<-matrix(c(2,3,11,5,6,7),nrow= 3, ncol= 2, byrow= TRUE)
> ##mat
> ##mat<-matrix(c(2,3,11,5,6,7),nrow= 2, ncol= 3, byrow= TRUE)
> ##mat
> x<-matrix(c(5:16), nrow=4, byrow= TRUE)
> y<-matrix(c(7:18),nrow=4, byrow= FALSE)
> x
  [,1] [,2] [,3]
[1,]   5   6   7
[2,]   8   9  10
[3,]  11  12  13
[4,]  14  15  16
> y
  [,1] [,2] [,3]
[1,]   7  11  15
[2,]   8  12  16
[3,]   9  13  17
[4,]  10  14  18
> ## Naming Matrix
> row_name<-c("r1","r2","r3","r4")
> col_names<-c("c1","c2","c3")
> z<- matrix(c(7:18), nrow=4, byrow= TRUE , dimnames= list(row_name,col_names))
> z
   c1 c2 c3
r1  7  8  9
r2 10 11 12
r3 13 14 15
r4 16 17 18
> ## Accessing elements
> print(z[3,1])
[1] 13
> print(z[,1])
r1 r2 r3 r4
 7 10 13 16
> print(z[2,])
c1 c2 c3
10 11 12
> ## updating elements
> z[4,3]<-0
> z[z==11]<-0
> z[z>15]<-0
> #cbind() and rbind() are used to add col n row resp
```

```

> rbind(z,c(2,3,4))
      c1 c2 c3
r1   7  8  9
r2  10  0 12
r3  13 14 15
r4   0  0  0
      2  3  4
> cbind(z,c(8,5,2,0))
      c1 c2 c3
r1   7  8  9  8
r2  10  0 12  5
r3  13 14 15  2
r4   0  0  0  0
> z
      c1 c2 c3
r1   7  8  9
r2  10  0 12
r3  13 14 15
r4   0  0  0
> ## Transpose i.e row into col n col into row
> t(z)
      r1 r2 r3 r4
c1   7 10 13  0
c2   8  0 14  0
c3   9 12 15  0
> a1<-matrix(c(5:16), nrow=4, ncol=3)
> a2<-matrix(c(1:12), nrow=4, ncol=3)
> sum<-a1+a2
> sum
      [,1] [,2] [,3]
[1,]    6   14   22
[2,]    8   16   24
[3,]   10   18   26
[4,]   12   20   28
> |

```

PRACTICAL NO 7

Aim - Implementation and analysis of Linear regression.

Load the dataset into R

```
dataset = read.csv("&quot;salary_data.csv&quot;")
```

Load the caTools library for splitting the dataset into training and test sets

```
library(caTools)
```

Split the dataset into training and test sets using a 2/3 to 1/3 ratio

```
split = sample.split(dataset$Salary, SplitRatio = 2/3)
```

Create the training set by subsetting the dataset where split is TRUE

```
training_set = subset(dataset, split == TRUE)
```

Create the test set by subsetting the dataset where split is FALSE

```
test_set = subset(dataset, split == FALSE)
```

Fit a linear regression model to the training data

Salary is the dependent variable, and YearsExperience is the independent variable

```
regressor = lm(formula = Salary ~ YearsExperience, data = training_set)
```

Summarize the fitted linear model to see the coefficients, R-squared value, and other statistics

```
summary(regressor)
```

Predict the salaries in the test set using the fitted linear model

```
salary_predict = predict(regressor, newdata = test_set)
```

```
# Print the actual and predicted salaries in the test set side by side
```

```
print(data.frame(Salary = test_set$Salary, Salary_Predict = salary_predict))
```

```
# Load the ggplot2 library for data visualization
```

```
library(ggplot2)
```

```
# Create a scatter plot with the training set points in green and test set points in red
```

```
# Add a regression line (fitted line) in blue based on the training set
```

```
ggplot() +
```

```
geom_point(aes(x = training_set$YearsExperience, y = training_set$Salary), colour = "#39;green#39;") +
```

```
geom_point(aes(x = test_set$YearsExperience, y = test_set$Salary), colour = "#39;red#39;") +
```

```
geom_line(aes(x = training_set$YearsExperience, y = predict(regressor, newdata = training_set)),  
colour = "#39;blue#39;") +
```

```
ggtitle("#39;Salary vs Experience (Green: Training Set, Red: Test Set)#39;") +
```

```
xlab("#39;Years of experience#39;") +
```

```
ylab("#39;Salary#39;")
```

```
# Load the dataset into R  
> dataset = read.csv("salary_data.csv")  
>  
> # Load the caTools library for splitting the dataset into training and test sets  
> library(caTools)  
>  
> # Split the dataset into training and test sets using a 2/3 to 1/3 ratio  
> split = sample.split(dataset$Salary, SplitRatio = 2/3)  
>  
> # Create the training set by subsetting the dataset where split is TRUE  
> training_set = subset(dataset, split == TRUE)  
>  
> # Create the test set by subsetting the dataset where split is FALSE  
> test_set = subset(dataset, split == FALSE)  
>  
> # Fit a linear regression model to the training data  
> # Salary is the dependent variable, and YearsExperience is the independent variable  
> regressor = lm(formula = Salary ~ YearsExperience, data = training_set)  
>  
> # Summarize the fitted linear model to see the coefficients, R-squared value, and other statistics  
> summary(regressor)
```

Call:

```
lm(formula = salary ~ YearsExperience, data = training_set)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-6621.7	-3157.1	-527.5	1790.9	12263.0

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	24916.0	2477.3	10.06	8.17e-09	***
YearsExperience	9460.2	427.3	22.14	1.65e-14	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5270 on 18 degrees of freedom

Multiple R-squared: 0.9646, Adjusted R-squared: 0.9626

F-statistic: 490.2 on 1 and 18 DF, p-value: 1.653e-14

```
>
> # Predict the salaries in the test set using the fitted linear model
> salary_predict = predict(regressor, newdata = test_set)
>
> # Print the actual and predicted salaries in the test set side by side
> print(data.frame(salary = test_set$salary, salary_predict = salary_predict))
  salary salary_predict
1  39343      35322.22
2  46205      37214.25
12 55794      62756.68
17 66029      73162.85
18 83088      75054.88
21 91738      89245.12
22 98273      92083.17
24 113812     102489.34
26 105582     110057.47
27 116969     114787.54
>
> # Load the ggplot2 library for data visualization
> library(ggplot2)
>
> # Create a scatter plot with the training set points in green and test set points in red
> # Add a regression line (fitted line) in blue based on the training set
> ggplot() +
+   geom_point(aes(x = training_set$YearsExperience, y = training_set$salary), colour = 'green') +
+   geom_point(aes(x = test_set$YearsExperience, y = test_set$salary), colour = 'red') +
+   geom_line(aes(x = training_set$YearsExperience, y = predict(regressor, newdata = training_set)), colour = 'blue') +
+   ggtitle('Salary vs Experience (Green: Training Set, Red: Test Set)') +
+   xlab('Years of experience') +
+   ylab('Salary')
> |
```



PRACTICAL NO 8

Aim - Implementation and Analysis Classification algorithms –Naïve Bayesian

```
install.packages("e1071")
install.packages("caTools")
install.packages("caret")
install.packages("ggplot2")
install.packages("lattice")

library(e1071)
library(caTools)
library(ggplot2)
library(lattice)
library(caret)

# splitting data into train and test data
View(iris)

split<-sample.split(iris, SplitRatio = 0.7)
train_cl<- subset(iris, split == "TRUE")
test_cl<- subset(iris, split == "FALSE")

# Feature Scaling
train_scale <-scale(train_cl[,1:4])
test_scale <- scale(test_cl[,1:4])

# Fitting Naive Bayes Model to Training data set
set.seed(120) # setting seed
classifier_cl <- naiveBayes(Species ~ .,data = train_cl)

classifier_cl

# predicting on test data
ypred <- predict(classifier_cl, newdata = test_cl)
```

```
# confusion Matrix
```

```
cm<- table(test_cl$Species,ypred)
```

```
cm
```

```
# Model Evaluation
```

```
confusionMatrix(cm)
```

```
> install.packages("e1071")
> install.packages("caTools")
> install.packages("caret")
> install.packages("ggplot2")
> install.packages("lattice")
> library(e1071)
> library(caTools)
> library(ggplot2)

> library(lattice)
> library(caret)
> # splitting data into train and test data
> view(iris)
> split<-sample.split(iris, splitRatio = 0.7)
> train_cl<- subset(iris, split == "TRUE")
> test_cl<- subset(iris, split == "FALSE")
> # Feature Scaling
> train_scale <-scale(train_cl[,1:4])
> test_scale <- scale(test_cl[,1:4])
> # Fitting Naive Bayes Model to Training data set
> set.seed(120) # setting seed
> classifier_cl <- naiveBayes(Species ~ .,data = train_cl)
> classifier_cl
```

Naive Bayes Classifier for Discrete Predictors

call:

```
naiveBayes.default(x = X, y = Y, laplace = laplace)
```

A-priori probabilities:

```
Y
  setosa versicolor virginica
0.3333333 0.3333333 0.3333333
```

Conditional probabilities:

```
      Sepal.Length
Y      [,1]      [,2]
setosa  4.986667 0.3812261
versicolor 5.950000 0.5556916
virginica 6.720000 0.5903827
```

```
      Sepal.width
Y      [,1]      [,2]
setosa  3.430000 0.3816028
versicolor 2.720000 0.3517444
virginica 2.983333 0.2730206
```

```
      Petal.Length
Y      [,1]      [,2]
setosa  1.440000 0.1693802
versicolor 4.283333 0.5239659
virginica 5.670000 0.5784343
```

```
      Petal.width
Y      [,1]      [,2]
setosa  0.2366667 0.1129032
versicolor 1.3100000 0.2171127
virginica 2.0500000 0.2403302
```

```
> # predicting on test data
> ypred <- predict(classifier_cl, newdata = test_cl)
> # confusion Matrix
> cm<- table(test_cl$Species,ypred)
> cm
```

	ypred		
	setosa	versicolor	virginica
setosa	20	0	0
versicolor	0	20	0
virginica	0	3	17

```
> # Model Evaluation
```

```
> confusionMatrix(cm)
```

Confusion Matrix and Statistics

	ypred		
	setosa	versicolor	virginica
setosa	20	0	0
versicolor	0	20	0
virginica	0	3	17

Overall Statistics

Accuracy : 0.95
 95% CI : (0.8608, 0.9896)
 No Information Rate : 0.3833
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.925

Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	0.8696	1.0000
Specificity	1.0000	1.0000	0.9302
Pos Pred Value	1.0000	1.0000	0.8500
Neg Pred Value	1.0000	0.9250	1.0000
Prevalence	0.3333	0.3833	0.2833
Detection Rate	0.3333	0.3333	0.2833
Detection Prevalence	0.3333	0.3333	0.3333
Balanced Accuracy	1.0000	0.9348	0.9651

```
> |
```

PRACTICAL NO 9

Aim - Implementation and Analysis Classification algorithms - K-Nearest Neighbour

```
install.packages("e1071")
```

```
install.packages("caTools")
```

```
install.packages("class")
```

```
# Load necessary libraries
```

```
# e1071 for machine learning functions, caTools for data splitting, and class for KNN
```

```
library(e1071)
```

```
library(caTools)
```

```
library(class)
```

```
# Load the built-in Iris dataset
```

```
data(iris) # Load the Iris dataset
```

```
head(iris) # Display the first few rows to understand the structure of the dataset
```

```
# Split the dataset into training and testing sets
```

```
set.seed(123) # Set a seed for reproducibility of random split
```

```
spl <- sample.split(iris, SplitRatio = 0.7) # Split data: 70% for training, 30% for testing
```

```
# Create training and testing subsets based on the split
```

```
train_cl <- subset(iris, spl == TRUE) # Training data
```

```
test_cl <- subset(iris, spl == FALSE) # Testing data
```

```
# Feature Scaling
```

```
# Standardize the numerical features to have mean 0 and standard deviation 1

train_scale <- scale(train_cl[, 1:4]) # Scale the first 4 columns (features) of training data

test_scale <- scale(test_cl[, 1:4]) # Scale the first 4 columns (features) of testing data


# Apply K-Nearest Neighbors (KNN) algorithm

# k = 1 means considering the nearest neighbor for classification

classifier_knn <- knn(train = train_scale, # Training data
                     test = test_scale,   # Testing data
                     cl = train_cl$Species, # Class labels for training data
                     k = 1)               # Number of neighbors to consider


# View the predictions for the testing data

print(classifier_knn)


# Create a confusion matrix to evaluate the model's performance

cm <- table(test_cl$Species, classifier_knn) # Compare actual vs predicted classes

print("Confusion Matrix:")

print(cm)


# Interpretation of the Confusion Matrix

# Rows represent the actual classes, and columns represent the predicted classes
```

```

R v. 4.5.2 . ~/
> install.packages("e1071")

WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate
version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/

Installing package into 'C:/Users/FYMCA/AppData/Local/R/win-library/4.5'
(as 'lib' is unspecified)

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.5/e1071_1.7-17.zip'

Content type 'application/zip' length 668407 bytes (652 KB)
downloaded 652 KB

package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\FYMCA\AppData\Local\Temp\Rtmp67IwX5\downloaded_packages
> install.packages("caTools")

WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate
version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/

Installing package into 'C:/Users/FYMCA/AppData/Local/R/win-library/4.5'
(as 'lib' is unspecified)

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.5/caTools_1.18.3.zip'

Content type 'application/zip' length 254350 bytes (248 KB)
downloaded 248 KB

package 'caTools' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\FYMCA\AppData\Local\Temp\Rtmp67IwX5\downloaded_packages
> install.packages("class")

WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate
version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/

Installing package into 'C:/Users/FYMCA/AppData/Local/R/win-library/4.5'
(as 'lib' is unspecified)

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.5/class_7.3-23.zip'

Content type 'application/zip' length 100243 bytes (97 KB)
downloaded 97 KB

package 'class' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\FYMCA\AppData\Local\Temp\Rtmp67IwX5\downloaded_packages
> # Load necessary libraries
> # e1071 for machine learning functions, caTools for data splitting, and class for KNN
>
> library(e1071)
> library(caTools)
> library(class)
> # Load the built-in Iris dataset
> data(iris) # Load the Iris dataset
> head(iris) # Display the first few rows to understand the structure of the dataset
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4          0.2  setosa
2          4.9         3.0          1.4          0.2  setosa
3          4.7         3.2          1.3          0.2  setosa
4          4.6         3.1          1.5          0.2  setosa
5          5.0         3.6          1.4          0.2  setosa
6          5.4         3.9          1.7          0.4  setosa
>

```

```

> # Split the dataset into training and testing sets
> set.seed(123) # Set a seed for reproducibility of random split
> spl <- sample.split(iris, SplitRatio = 0.7) # Split data: 70% for training, 30% for testing
> # Create training and testing subsets based on the split
> train_cl <- subset(iris, spl == TRUE) # Training data
> test_cl <- subset(iris, spl == FALSE) # Testing data
>
> # Feature Scaling
> # Standardize the numerical features to have mean 0 and standard deviation 1
> train_scale <- scale(train_cl[, 1:4]) # Scale the first 4 columns (features) of training data
> test_scale <- scale(test_cl[, 1:4]) # Scale the first 4 columns (features) of testing data
> # Apply K-Nearest Neighbors (KNN) algorithm
> # k = 1 means considering the nearest neighbor for classification
> classifier_knn <- knn(train = train_scale, # Training data
+                       test = test_scale, # Testing data
+                       cl = train_cl$species, # Class labels for training data
+                       k = 1) # Number of neighbors to consider
>
> # View the predictions for the testing data
> print(classifier_knn)
[1] setosa setosa setosa setosa setosa setosa setosa setosa setosa
[11] setosa setosa setosa setosa setosa setosa setosa setosa setosa
[21] versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[31] versicolor versicolor virginica versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[41] virginica virginica virginica virginica virginica virginica virginica versicolor virginica virginica
[51] virginica virginica versicolor versicolor virginica virginica virginica virginica virginica virginica
Levels: setosa versicolor virginica
> # Create a confusion matrix to evaluate the model's performance
> cm <- table(test_cl$species, classifier_knn) # Compare actual vs predicted classes
> print("Confusion Matrix:")
[1] "Confusion Matrix:"
> print(cm)
      classifier_knn
      setosa versicolor virginica
setosa      20         0         0
versicolor   0        19         1
virginica    0         3        17
>
> # Interpretation of the Confusion Matrix
> # Rows represent the actual classes, and columns represent the predicted classes
> |

```


PRACTICAL NO 10

Aim - Implementation and Analysis Classification algorithms - ID3, C4.5

```
# Load necessary libraries
install.packages("RWeka")
library(RWeka)

# Read the CSV file
data <- read.csv("weight-height.csv")

# Convert the class attribute to a factor
data$Gender <- as.factor(data$Gender)

# Build the C4.5 (J48) model
model_c45 <- J48(Gender ~ Height + Weight, data = data)

# Print the model
print(model_c45)

# Predict using the model
a <- data.frame(Height = c(190), Weight = c(85))
result <- predict(model_c45, a)
print(result)

# Visualization (J48 does not have a straightforward plot method like rpart)
# summary of the model
summary(model_c45)
```

```
> library(Rweka)
>
> # Read the CSV file
> data <- read.csv("weight-height.csv")
>
> # Convert the class attribute to a factor
> data$Gender <- as.factor(data$Gender)
>
> # Build the C4.5 (J48) model
> model_c45 <- J48(Gender ~ Height + weight, data = data)
>
> # Print the model
> print(model_c45)
```

J48 pruned tree

```
-----  
weight <= 162.676838  
| weight <= 145.689533: Female (3592.0/91.0)  
| weight > 145.689533  
| | weight <= 156.274297  
| | | Height <= 63.186865  
| | | | Height <= 61.368045: Male (9.0)  
| | | | Height > 61.368045  
| | | | | weight <= 148.832458: Female (41.0/12.0)  
| | | | | weight > 148.832458: Male (44.0/17.0)  
| | | | Height > 63.186865: Female (882.0/144.0)  
| | | weight > 156.274297  
| | | | Height <= 65.605575: Male (204.0/64.0)  
| | | | Height > 65.605575: Female (363.0/103.0)  
weight > 162.676838  
| weight <= 179.181924  
| | weight <= 169.128653  
| | | Height <= 68.068922: Male (487.0/136.0)  
| | | Height > 68.068922: Female (107.0/34.0)  
| | weight > 169.128653: Male (964.0/132.0)  
| weight > 179.181924: Male (3307.0/50.0)
```

Number of Leaves : 11

Size of the tree : 21

```
>  
> # Predict using the model  
> a <- data.frame(Height = c(190), weight = c(85))  
> result <- predict(model_c45, a)  
> print(result)  
[1] Female  
Levels: Female Male  
>  
> # visualization (J48 does not have a straightforward plot method like rpart)  
> # summary of the model  
> summary(model_c45)
```

=== Summary ===

Correctly Classified Instances	9217	92.17	%
Incorrectly Classified Instances	783	7.83	%
Kappa statistic	0.8434		
Mean absolute error	0.126		
Root mean squared error	0.251		
Relative absolute error	25.2074	%	
Root relative squared error	50.207	%	
Total Number of Instances	10000		

=== Confusion Matrix ===

```
   a    b  <-- classified as  
4601 399 |    a = Female  
 384 4616 |    b = Male  
> |
```

PRACTICAL NO 11

Aim - Implementation and analysis of Apriori Algorithm using Market Basket Analysis

Install the arules package for association rule mining

```
install.packages("arules")
```

Load the arules library

```
library(arules)
```

Load the Groceries dataset from the arules package

```
data("Groceries")
```

Display the attributes of the Groceries dataset

```
attributes(Groceries)
```

Inspect the first 3 transactions in the Groceries dataset

```
inspect(head(Groceries, 3))
```

Open the Groceries dataset in a spreadsheet-like viewer

```
View(Groceries)
```

Display a summary of the Groceries dataset

```
summary(Groceries)
```

Create an Apriori model with a minimum support of 0.001 and confidence of 0.15

```
model <- apriori(data = Groceries, parameter = list(support = 0.001, confidence = 0.15))
```

```
# Inspect the first 3 association rules generated by the Apriori model
```

```
inspect(head(model, 3))
```

```
> # Install the arules package for association rule mining
```

```
> install.packages("arules")
```

```
> # Load the arules library
```

```
> library(arules)
```

```
> # Load the Groceries dataset from the arules package
```

```
> data("Groceries")
```

```
> # Display the attributes of the Groceries dataset
```

```
> attributes(Groceries)
```

```
$data
```

```
169 x 9835 sparse Matrix of class "ngcMatrix"
```

```
[1,] .....|.....
[2,] .....|.....|.....
[3,] .....
[4,] .....|.....
[5,] .....
[6,] .....
[7,] .....
[8,] .....|.....
[9,] .....
```

```
.....
```

```
.....suppressing 9782 columns and 151 rows in show(); maybe adjust options(max.print=, width=)
```

```
.....
```

155	cookware	non-food kitchen	non-food
156	kitchen utensil	non-food kitchen	non-food
157	cling film/bags	non-food kitchen	non-food
158	kitchen towels	non-food house keeping products	non-food
159	house keeping products	non-food house keeping products	non-food
160	candles	non-food house keeping products	non-food
161	light bulbs	non-food house keeping products	non-food
162	sound storage medium	games/books/hobby	non-food
163	newspapers	games/books/hobby	non-food
164	photo/film	games/books/hobby	non-food
165	pot plants	garden	non-food
166	flower soil/fertilizer	garden	non-food
167	flower (seeds)	garden	non-food
168	shopping bags	bags	non-food
169	bags	bags	non-food

```
$itemsetInfo
```

```
data frame with 0 columns and 0 rows
```

```
$class
```

```
[1] "transactions"
```

```
attr(,"package")
```

```
[1] "arules"
```

```
> # Inspect the first 3 transactions in the Groceries dataset
```

```
> inspect(head(Groceries, 3))
```

```
items
```

```
[1] {citrus fruit, semi-finished bread, margarine, ready soups}
```

```
[2] {tropical fruit, yogurt, coffee}
```

```
[3] {whole milk}
```

```
> # Open the Groceries dataset in a spreadsheet-like viewer
```

```
> view(Groceries)
```

```
> # Display a summary of the Groceries dataset
```

```
> summary(Groceries)
```

```
transactions as itemMatrix in sparse format with
9835 rows (elements/itemsets/transactions) and
169 columns (items) and a density of 0.02609146
```

```
most frequent items:
```

whole milk	other vegetables	rolls/buns	soda	yogurt	(Other)
2513	1903	1809	1715	1372	34055

```
element (itemset/transaction) length distribution:
```

```
sizes
 1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24
2159 1643 1299 1005  855  645  545  438  350  246  182  117  78   77   55   46   29   14   14    9   11    4    6    1
 26   27   28   29   32
 1    1    1    3    1

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.000  2.000   3.000   4.409   6.000   32.000
```

```
includes extended item information - examples:
```

labels	level2	level1
1 frankfurter	sausage	meat and sausage
2 sausage	sausage	meat and sausage
3 liver loaf	sausage	meat and sausage

```
> # Create an Apriori model with a minimum support of 0.001 and confidence of 0.15
> model <- apriori(data = Groceries, parameter = list(support = 0.001, confidence = 0.15))
Apriori
```

```
Parameter specification:
```

confidence	minval	smax	aref	aval	originalsupport	maxtime	support	minlen	maxlen	target	ext
0.15	0.1	1	none	FALSE	TRUE	5	0.001	1	10	rules	TRUE

```
Algorithmic control:
```

filter	tree	heap	memopt	load	sort	verbose
0.1	TRUE	TRUE	FALSE	TRUE	2	TRUE

```
Absolute minimum support count: 9
```

```
set item appearances ... [0 item(s)] done [0.00s].
set transactions ... [169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.01s].
writing ... [26824 rule(s)] done [0.06s].
creating S4 object ... done [0.01s].
```

```
> # Inspect the first 3 association rules generated by the Apriori model
> inspect(head(model, 3))
```

	lhs	rhs	support	confidence	coverage	lift	count
[1]	{}	=> {soda}	0.1743772	0.1743772	1	1	1715
[2]	{}	=> {rolls/buns}	0.1839349	0.1839349	1	1	1809
[3]	{}	=> {other vegetables}	0.1934926	0.1934926	1	1	1903

PRACTICAL NO 12

Aim - Implementation and analysis of clustering algorithms - K-Means

Install necessary packages

install.packages("stats") # For statistical operations

install.packages("dplyr") # For data manipulation

install.packages("ggplot2") # For data visualization

install.packages("ggfortify") # For ggplot2-based plotting of statistical results

Load the libraries

library(stats) # Load stats package

library(dplyr) # Load dplyr package for data manipulation

library(ggplot2) # Load ggplot2 package for data visualization

library(ggfortify) # Load ggfortify package for enhanced ggplot2 functionality

View the built-in iris dataset

View(iris)

Select the first four columns (sepal length, sepal width, petal length, petal width) from the iris dataset

mydata = select(iris, c(1, 2, 3, 4))

Perform K-means clustering on the selected data with 2 clusters

km = kmeans(mydata, 2)

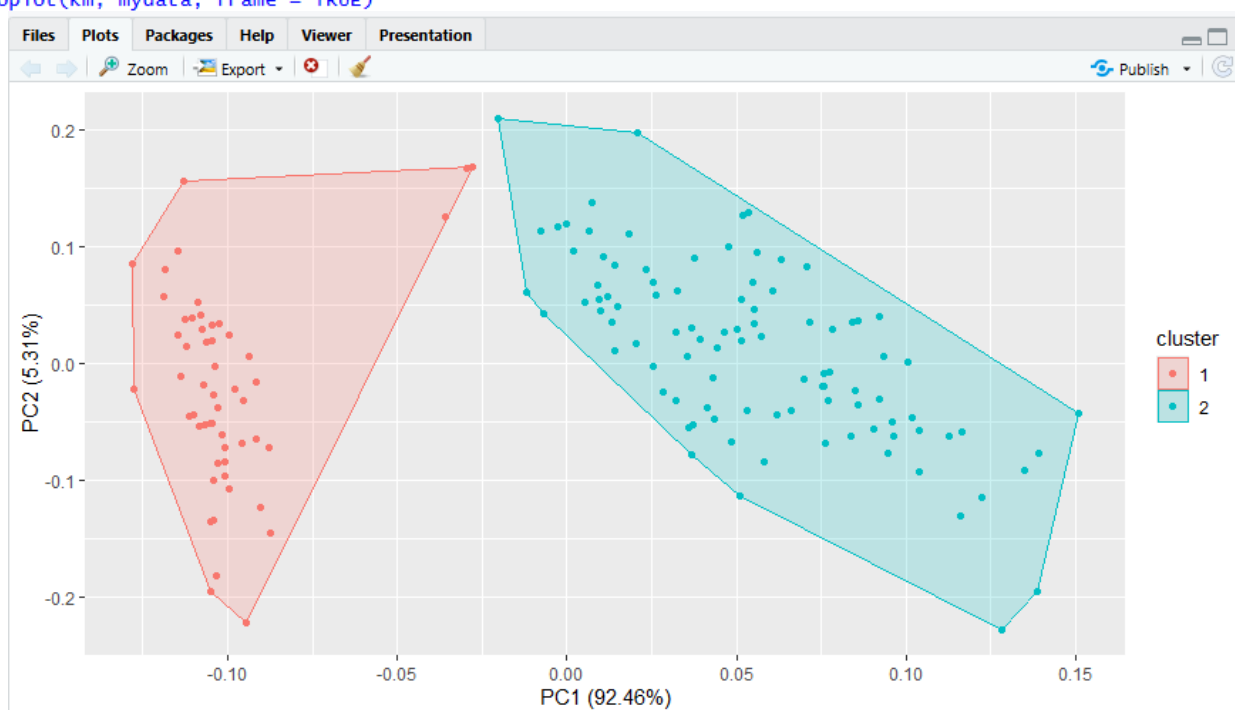
Plot the clustering results using autoplot from ggfortify, with frames around clusters

autoplot(km, mydata, frame = TRUE)

```

> # Install necessary packages
> install.packages("stats")      # For statistical operations
> install.packages("dplyr")
> install.packages("ggplot2")    # For data visualization
> install.packages("ggfortify")  # For ggplot2-based plotting of statistical results
> # Load the libraries
> library(stats)                # Load stats package
> library(dplyr)                # Load dplyr package for data manipulation
> library(ggplot2)              # Load ggplot2 package for data visualization
> library(ggfortify)            # Load ggfortify package for enhanced ggplot2 functionality
> # View the built-in iris dataset
> view(iris)
> # Select the first four columns (sepal length, sepal width, petal length, petal width) from the iris dataset
> mydata = select(iris, c(1, 2, 3, 4))
> # Perform K-means clustering on the selected data with 2 clusters
> km = kmeans(mydata, 2)
> # Plot the clustering results using autoplot from ggfortify, with frames around clusters
> autoplot(km, mydata, frame = TRUE)

```



PRACTICAL NO 13

Aim - Implementation and analysis of clustering algorithms - Agglomerative

Step 1: Load necessary libraries

```
library(datasets) # To access the iris dataset
```

```
library(ggplot2) # For visualization
```

Step 2: Load and explore the dataset

```
data(iris) # Load the iris dataset
```

View the first few rows of the dataset

```
head(iris)
```

Step 3: Preprocess the Data

Exclude the Species column and normalize the data

```
iris_data <- iris[, -5] # Remove the Species column
```

```
iris_scaled <- scale(iris_data) # Normalize the features to have mean 0 and standard deviation 1
```

Step 4: Perform Agglomerative Clustering

Compute the distance matrix (Euclidean distance)

```
dist_matrix <- dist(iris_scaled, method = "euclidean")
```

Perform hierarchical clustering using complete linkage

```
agg_clust <- hclust(dist_matrix, method = "complete")
```

Step 5: Visualize the Dendrogram

Plot the dendrogram to see how clusters are formed

```
plot(agg_clust, main = "Dendrogram of Agglomerative Clustering", xlab = "", sub = "", cex = 0.9)
```

```
# Step 6: Cut the Dendrogram to form 3 clusters
```

```
# Define the number of clusters as 3
```

```
clusters <- cutree(agg_clust, k = 3)
```

```
# Add the cluster labels to the iris dataset
```

```
iris$Cluster <- as.factor(clusters)
```

```
# View the first few rows of the dataset with cluster assignments
```

```
head(iris)
```

```
# Step 7: Visualize the Clusters
```

```
# Scatter plot using ggplot2 to visualize the clusters based on Sepal.Length and Sepal.Width
```

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Cluster)) +
```

```
  geom_point(size = 3) +
```

```
  labs(title = "Agglomerative Clustering - Clustered Data",
```

```
        x = "Sepal Length",
```

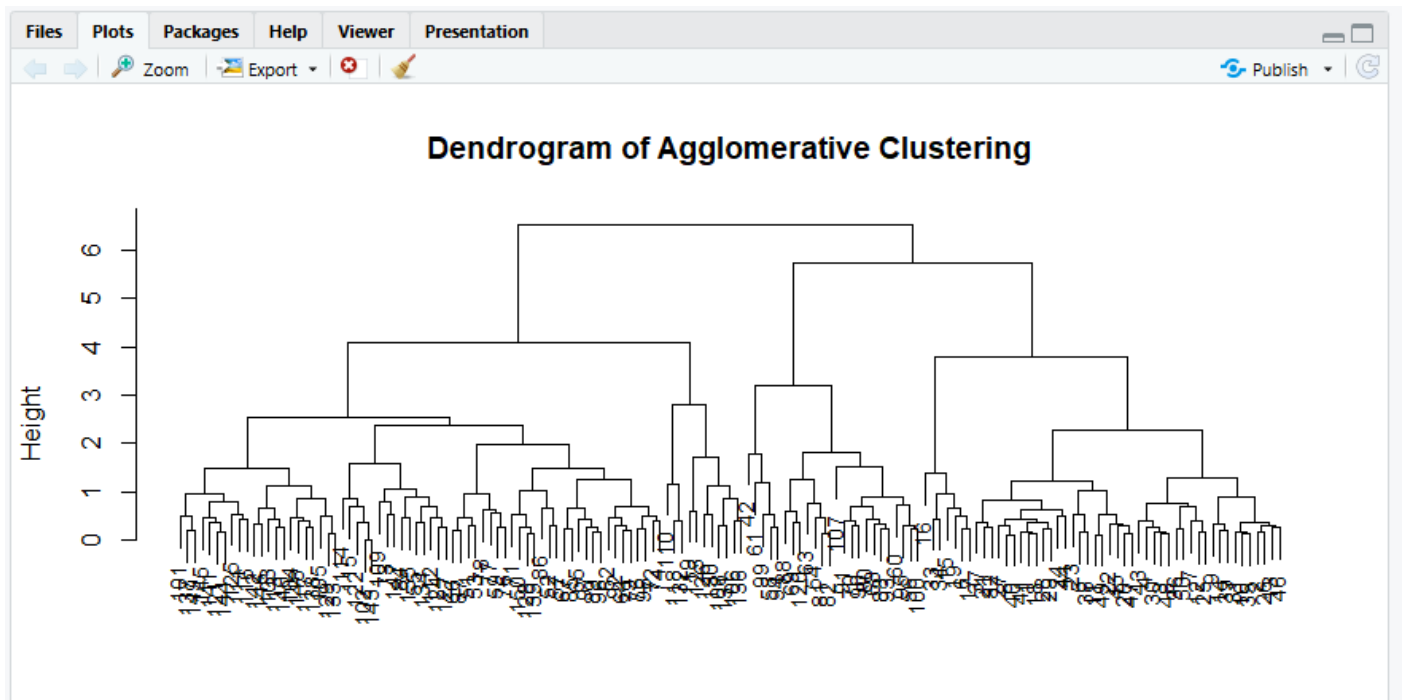
```
        y = "Sepal Width") +
```

```
  theme_minimal()
```

```

> # Step 1: Load necessary libraries
> library(datasets) # To access the iris dataset
> library(ggplot2) # For visualization
> # Step 2: Load and explore the dataset
> data(iris) # Load the iris dataset
> # View the first few rows of the dataset
> head(iris)
  Sepal.Length Sepal.width Petal.Length Petal.width Species
1          5.1         3.5         1.4         0.2  setosa
2          4.9         3.0         1.4         0.2  setosa
3          4.7         3.2         1.3         0.2  setosa
4          4.6         3.1         1.5         0.2  setosa
5          5.0         3.6         1.4         0.2  setosa
6          5.4         3.9         1.7         0.4  setosa
> # Step 3: Preprocess the Data
> # Exclude the Species column and normalize the data
> iris_data <- iris[, -5] # Remove the Species column
> iris_scaled <- scale(iris_data) # Normalize the features to have mean 0 and standard deviation 1
> # Step 4: Perform Agglomerative Clustering
> # Compute the distance matrix (Euclidean distance)
> dist_matrix <- dist(iris_scaled, method = "euclidean")
> # Perform hierarchical clustering using complete linkage
> agg_clust <- hclust(dist_matrix, method = "complete")
> # Step 5: visualize the Dendrogram
> # Plot the dendrogram to see how clusters are formed
> plot(agg_clust, main = "Dendrogram of Agglomerative Clustering", xlab = "", sub = "", cex = 0.9)
>

```



```

> # Step 6: Cut the Dendrogram to form 3 clusters
> # Define the number of clusters as 3
> clusters <- cutree(agg_clust, k = 3)
> # Add the cluster labels to the iris dataset
> iris$cluster <- as.factor(clusters)
> # View the first few rows of the dataset with cluster assignments
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Cluster
1          5.1         3.5         1.4         0.2   setosa        1
2          4.9         3.0         1.4         0.2   setosa        1
3          4.7         3.2         1.3         0.2   setosa        1
4          4.6         3.1         1.5         0.2   setosa        1
5          5.0         3.6         1.4         0.2   setosa        1
6          5.4         3.9         1.7         0.4   setosa        1
> # Step 7: Visualize the clusters
> # Scatter plot using ggplot2 to visualize the clusters based on Sepal.Length and Sepal.Width
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = cluster)) +
+   geom_point(size = 3) +
+   labs(title = "Agglomerative Clustering - Clustered Data",
+        x = "Sepal Length",
+        y = "Sepal Width") +
+   theme_minimal()

```

